

3日^{3days master}で
マスター[®]

3日あれば
十分にやば〜

JS

Java Script

伊藤静香 著

短期集中レッスン!

プログラマでも、Webデザイナーでも、
必ず覚えておきたい基礎知識を
ギュッと詰め込んだ入門書です。

1 Day

2 Day

3 Day

ソエム

3日^{3days master}で
マスター[®]

JavaScript

伊藤静香 著

Microsoft、Windows、Windows 7、Internet Explorer は Microsoft Corporation の米国および各国における商標または登録商標です。

Apple、Macintosh、Power Mac、Mac、Mac OS、Mac OS X、QuickTime は Apple Computer, Inc. の米国および各国における商標または登録商標です。

登録商標「3日でマスター」(第 5330381 号)は、ソシム株式会社が本商品に使用する許諾を得ています。その他、本書に掲載されているすべてのブランド名と製品名、商標または登録商標は、それぞれ帰属者の所有物です。

本書中に ®、©、™ は明記していません。

- 本書はソシム株式会社が出版したもので、本書に関する権利、責任はソシム株式会社が保有します。
- 本書のいかなる部分についても、ソシム株式会社との書面による事前の同意なしに、電気、機械、複写、録音、その他のいかなる形式や手段によっても、複製、および検索システムへの保存や転送は禁止されています。
- 本書の内容は参照用としてのみ使用されるべきものであり、予告なしに変更されることがあります。また、ソシム株式会社がその内容を保証するものではありません。本書の内容に誤りや不正確な記述がある場合も、ソシム株式会社はその一切の責任を負いません。
- 本書に記載されている内容の運用によって、いかなる損害が生じても、ソシム株式会社および著者は責任を負いかねますので、あらかじめご了承ください。

まえがき



継続は力なり。

何か新しいことを始めたり、身に付けたり、

物事を成し遂げたりするためには、

少しずつでも続けることが何より大事。

多くの先人、偉人、成功者が異口同音に挙げているコツの1つです。

しかし、続けることは、単純だけど難しい。

やり始めるのは簡単ですが、

毎日コツコツと続けることは、思う以上にハードルの高い行為です。

何度となく「3日坊主」になってしまった経験を持つ人は多いのではないのでしょうか。

ただ、これを裏返して、ポジティブに考えれば、

どんなに飽きっぽくて、根気のない人でも、

「最低3日なら続けられる」とも言えます。

だったら、その3日のうちに、

必要最低限のことをマスターしてしまえばいいのではないか？

この本は、そのようなコンセプトで作られています。

この本では、JavaScriptの基本的な文法と使い方を

3日間でマスターし、

次のようなことができるところまでを目指します。

- ゼロからごくごく簡単なJavaScriptプログラムを作れる
- 既存のJavaScriptプログラムを見て、何が書いてあるか、だいたいわかる
- より専門的で高度な参考書や記事を読んで理解できる

JavaScript とは？

JavaScriptは現在、もっとも広く使われているプログラミング言語の1つです。

私たちがふだん目にするほとんどのWebサイト、Webページで使われています。

10～15年ぐらい前までは、JavaScript のプログラミングは、
もっぱらプログラマの仕事でした。
しかし今日では、プログラマだけでなく、
Web デザイナーや Web 担当者も
JavaScript の基礎知識ぐらいは知っておかなくては仕事にならない、
という状況になってきています。

この本が対象としている読者

この本は、おもに次のような読者を対象にしています。

- 基礎的な HTML と CSS の知識はある人
- プログラミングというものをやったことがない人
- JavaScript を見よう見まねで触ってきたけれど、ちゃんと勉強したことはない人

もちろん、他のプログラミング言語でプログラミングをしたことはあるけれど、
JavaScript は初めて、という方もいらっしゃるかと思います。
そういう人には、ちょっと基本的で簡単すぎるかもしれませんが、
もちろん大歓迎です。
(念のため、ちょっと立ち読みしていただいて、
簡単すぎるようなら、そっと本棚にお戻しください。)

あらかじめ HTML と CSS の知識が必要

JavaScript のプログラムは、
プログラムの中で HTML や CSS のソースコードを操作することで、
文字や画像などのグラフィカルな動きを実現しています。
ですので、JavaScript でプログラムを作成するには、
HTML の知識と CSS の知識も必要となります。

本書では、HTML と CSS についてはすでに基本的な知識があるという前提で、
JavaScript に絞って説明を進めていきます。

HTML や CSS がわからないという方は、
あらかじめ『3日でマスター HTML5 & CSS3』（ソシム刊）など
HTML と CSS を解説している本で、
基本をマスターしてから読み始めてください。



注意

この本での説明には、実用的なサンプルは使っていません。
サンプルは、文法内容を理解してもらうためと割り切って、
できるだけ余計な要素をそぎ落とした、ごく短いサンプルにしてあります。
ですので、すぐに Web サイトで使えるような実用的なサンプルで学びたい方には、
不向きな本となっています。

また、プログラミング未経験者にもわかりやすいように、
初心者の段階ではあまり知る必要のない細かい補足は省き、
言葉の定義については、できるだけ専門用語を使わないようにしています。
くどいところがあったり、厳密さを欠くところがあったりするかもしれませんが、
そうした点は、この本のあとで、別の詳しい本でしっかり学習してください。

学習時間のめやす

3 日でマスターするとはいっても、
1 日 10 時間も 15 時間も勉強する必要はありません。
この本では、以下のような想定で、分量の上限を設けました。

- 学習時間：1 日あたり 2 時間程度
- ページ数：1 日あたり 60 ページ以内



1 ページを平均 2 分の速度で読めれば、
1 時間で 30 ページ、2 時間で 60 ページ読める計算になります。
1 日 2 時間を 3 日間続けることで、
JavaScript プログラミングに必要と思われる基本的な知識を
効率よくマスターできるように、構成してあります。

はじめて JavaScript を学ぶ方はもちろん、
これまでに始めたことがあるけれど途中で挫折してしまった人も、
ぜひ気軽に始めてみてください。
みなさんの出発（または再出発）のお役に立てれば幸いです。

伊藤 静香

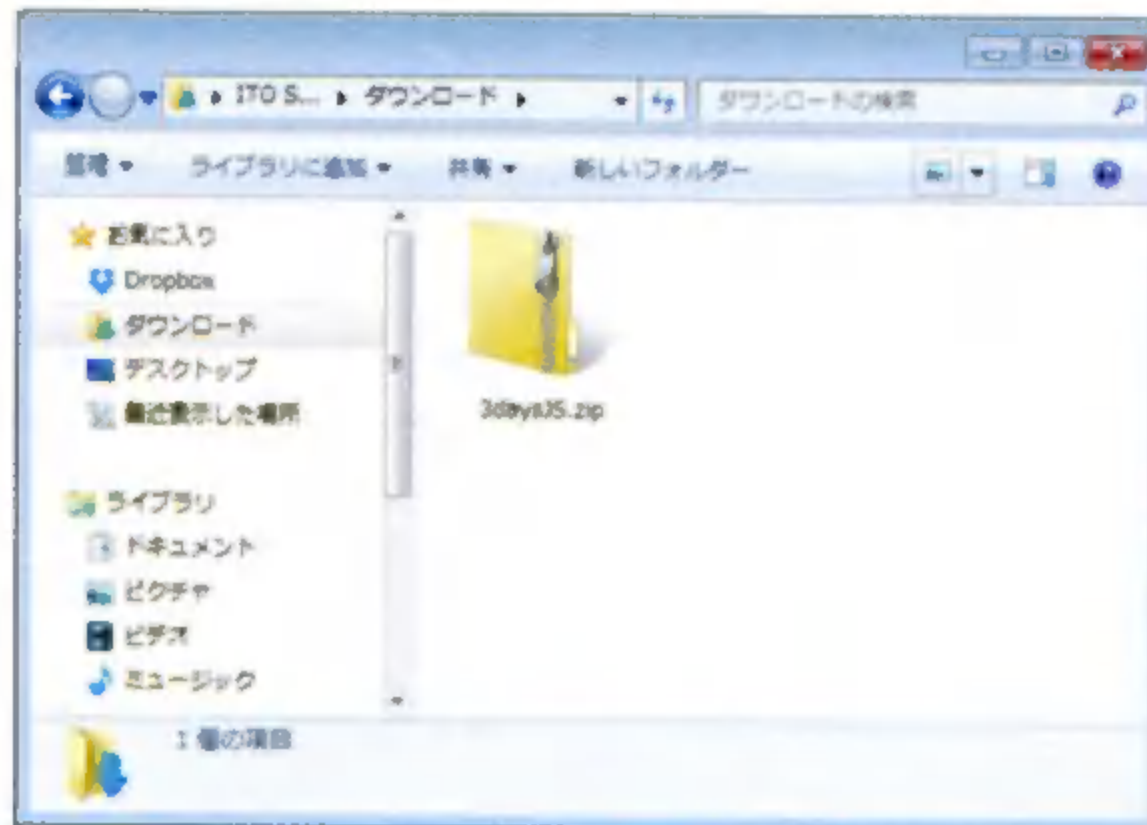
●動作確認用サンプルプログラムをダウンロードする

この本で作るサンプルプログラムは、以下のURLからダウンロードできます。
サンプルコードをいちいち自分で入力しなくても、ブラウザで開けばすぐに実際の動作が確認できますし、自分で入力したファイルが思いどおりに表示されないときに、比較して確かめることもできます。

〈サンプルプログラムのダウンロード〉

<http://www.socym.co.jp/book/934/>

※読者サポートの「ダウンロード」よりダウンロードしてください。



ダウンロードされた
サンプルプログラム

ダウンロードしたファイルは、通常「ダウンロード」フォルダの中にダウンロードされています。解凍して、自分の使いやすいフォルダ（「デスクトップ」や「ドキュメント」）へ移動しておいてください。

動作環境

この本で示している操作手順や、ブラウザでの表示例には、以下の環境を使用しています。

- ・ Windows 7
- ・ Google Chrome バージョン 35

Windows XP や Windows Vista、Windows 8 といった他の Windows OS や、Mac OS X、また Google Chrome 以外のブラウザでは、操作手順や画面表示に多少違いがある場合があります。



CONTENTS

Day1

Lesson 1 プログラム学習のための準備をする 11

- 1 JavaScriptでどんなことができるのか? 12
- 2 プログラム作成に必要なソフト 14
- 3 ブラウザをインストールする 16
- 4 テキストエディタを起動する 17

Lesson 2 1行だけのプログラムを作ってみる 19

- 1 ソースコードを書く 20
- 2 プログラムを実行する 22
- 3 プログラムを書き換えて実行する 25
- 4 ファイルの拡張子が見えるように設定する 27

Lesson 3 プログラム作成前に必要な知識 31

- 1 プログラムとは何か? 32
- 2 ソースコードの書き方 35

Lesson 4 変数のしくみと単純なデータ 37

- 1 変数の使い方 38
- 2 データの種類①——数値 41
- 3 データの種類②——文字列 42
- 4 データの種類③——論理値 44
- 5 算術演算子 45
- 6 連結演算子 46
- 7 コメントの書き方 47
- 8 データ型 48

Lesson 5 データの種類④——配列.....49

1 変数に複数のデータを代入する方法	50
2 配列のしくみ	51
3 先頭の要素を削除する shift() メソッド	55
4 先頭に要素を追加する unshift() メソッド	56
5 末尾の要素を削除する pop() メソッド	57
6 末尾に要素を追加する push() メソッド	58
7 要素を削除する splice() メソッド	59
8 配列の長さを調べる length プロパティ	60

Lesson 6 データの種類⑤——オブジェクト61

1 オブジェクトのしくみ	62
2 オブジェクトの使い方	66
3 2つの特殊なデータ型	68

Day2

Lesson 1 条件文69

1 条件文①——if 文	70
2 条件文②——if ~ else 文	72
3 条件文③——if ~ else if ~ else 文	74
4 条件文④——switch 文	77
5 条件の書き方	79

Lesson 2 繰り返し文81

1 繰り返し文①——while 文	82
2 繰り返し文②——for 文	86
3 繰り返し文③——do ~ while 文	88
4 for 文と配列を使った繰り返し処理	90
5 インクリメント演算子とデクリメント演算子	92

Lesson 3 関数 93

1 関数のしくみ	94
2 戻り値を出力する関数	96
3 引数を受け取る関数	99
4 変数のスコープ	102
5 関数の定義方法の3つのバリエーション	103

Lesson 4 組み込みオブジェクト 105

1 組み込みオブジェクトのしくみ	106
2 Dateオブジェクト	108
3 Mathオブジェクト	111
4 RegExpオブジェクト	113
5 Arrayオブジェクト	117
6 Stringオブジェクト	119
7 Numberオブジェクト	121
8 Functionオブジェクト	122
9 その他の組み込みオブジェクト	123

Day3

Lesson 1 windowオブジェクトを使って ブラウザを操作する 125

1 windowオブジェクト	126
2 alert()メソッドなど	128
3 open()メソッド	129
4 innerWidthプロパティなど	131
5 locationオブジェクト	133
6 historyオブジェクト	135
7 navigatorオブジェクト	136
8 documentオブジェクト	137

Lesson 2 DOMの基本的な使い方 139

- 1 DOMとは何か? 140
- 2 要素名で要素を取得する 141
- 3 要素の内容を取得する 143
- 4 要素の内容を変更する 144
- 5 id属性の値にマッチする要素を取得する 146
- 6 class属性の値にマッチする要素を取得する 148

Lesson 3 イベントとイベントハンドラ 149


- 1 イベントとは何か? 150
- 2 イベントハンドラの使い方 151
- 3 開始タグの中で関連付ける方法 152
- 4 プロパティを使って関連付ける方法 154
- 5 loadイベントの使い方 156
- 6 addEventListener() メソッド 159
- 7 タッチデバイスのイベント 160

Lesson 4 jQueryの使い方 161

- 1 jQueryとは何か? 162
- 2 jQueryをインストールする 163
- 3 jQueryを読み込む設定をする 165
- 4 jQueryの記述方法 166
- 5 css()メソッドで要素の色を変える 168
- 6 html()メソッドで要素の中身を変える 170
- 7 fadeOut()メソッドでアニメーション効果 171
- 8 メソッドを繋げて書くメソッドチェーン 172
- 9 イベント処理を設定する 173

索引 174

Day 1



Lesson 1

プログラム学習のための 準備をする

このレッスンでは、JavaScriptのプログラムを作成するために必要なソフトの準備をします。

- 1 JavaScriptでどんなことができるのか？
- 2 プログラム作成に必要なソフト
- 3 ブラウザをインストールする
- 4 テキストエディタを起動する

JavaScriptでどんなことができるのか？

SECTION

1

多くの Web サイトでよく使われている機能を例に、代表的な JavaScript プログラムをいくつか紹介しておきます。

● ブラウザで動作するプログラムを作れる

JavaScriptは、ブラウザで動くプログラムが作れるプログラミング言語です。Web ページの見た目を動的に変化させたり、ユーザーのマウスやキーの操作に対応するような動きを付けたりすることができます。よく使われている例としては、たとえば、オートコンプリート、ドロップダウンメニュー、スライドショーなどといった機能があります。

● オートコンプリート

Googleの検索ボックスに文字を入力すると、1文字入力するごとに、検索キーワードの候補が表示されます。このような機能をオートコンプリートといいます。JavaScriptで作られていることが多いです。

Google (<https://www.google.co.jp/>)



検索キーワード候補が表示される

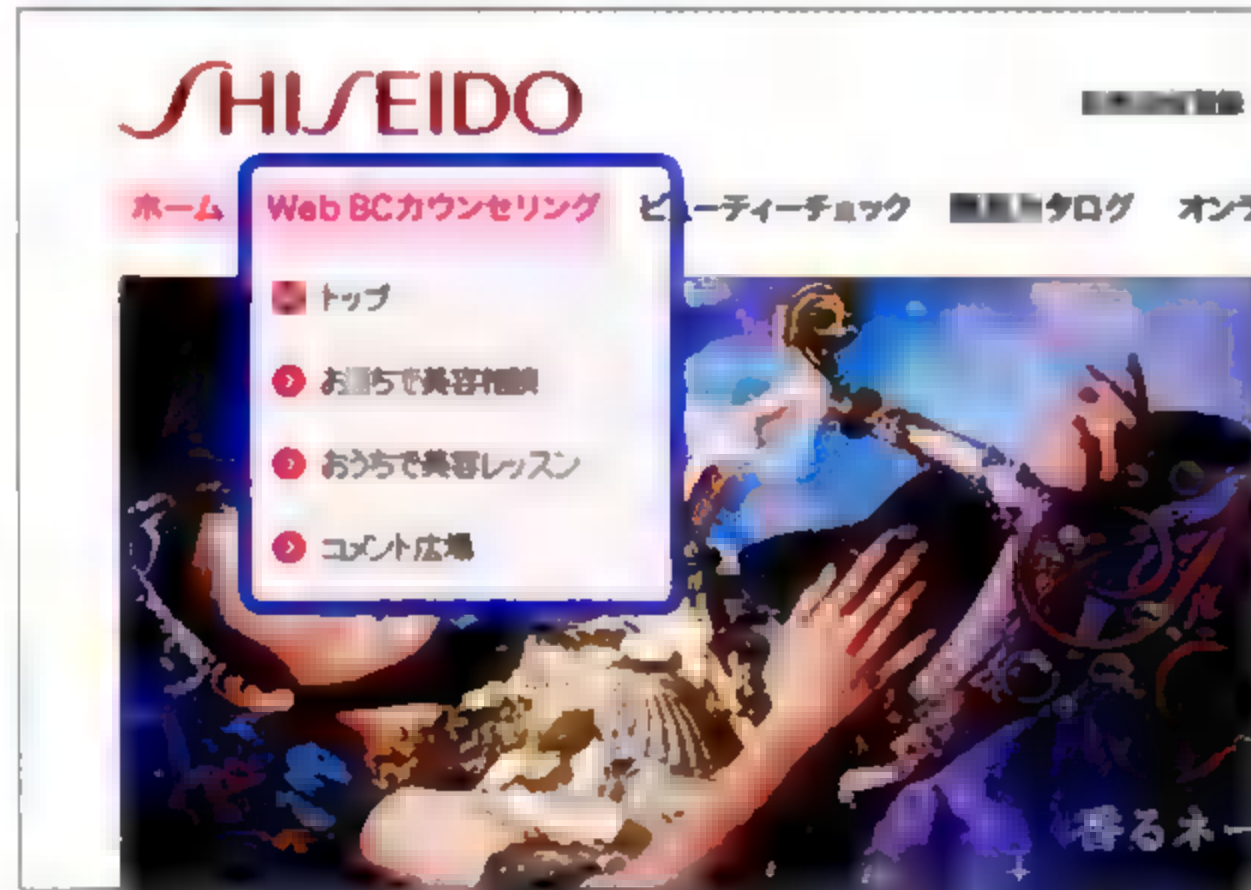
この画面では、「東京」と入力すると、その下に「天気予報 東京」「東京電力」「東京メトロ」「東京駅」などの検索キーワード候補が表示されています。



● ドロップダウンメニュー

メニューをクリックしたり、マウスポインタを重ねたりしたときに表示されるドロップダウンメニューという機能がありますが、これもまたJavaScriptによって作られていることが多い機能です。

▼ ワタシプラス／資生堂 (<http://www.shiseido.co.jp/>)



メニューリストが表示される

● スライドショー

画像をクリックすると、そのWebページの中で新しい画像が次々に表示されるしくみをスライドショーといいます。JavaScriptを使えば、このスライドショーのような機能も実現可能です。

▼ 東京ガス (<http://www.tokyo-gas.co.jp/>)



新しい画像が次々に表示される



プログラム作成に必要なソフト

JavaScript のプログラムを作るには、テキストエディタと Web ブラウザが必要です。どちらも無料で手に入ります。

● プログラムを作るために必要な 2 つのソフト

JavaScript のプログラムを作るには、2 つのソフトが必要です。1 つは、プログラムのソースコード（中身のこと）を書くための**テキストエディタ**。もう1 つは、作ったプログラムが正しく動くかどうかを確認する**Web ブラウザ**です。

● テキストエディタとは？

テキストエディタ（単にエディタともいう）とは、**文字入力に特化したソフト**のことです。Windows パソコンには、最初から「メモ帳」というソフトが入っていますが、この「メモ帳」はテキストエディタの1 つです。

また、Mac のパソコンであれば、「テキストエディット」というテキストエディタが入っています。

テキストエディタには有料無料含めて、たくさんのソフトがありますので、「メモ帳」以外のテキストエディタをお使いいただいてもかまいません。この本では、Windows の「メモ帳」を使って説明していきます。

MEMO

「Word」や「一太郎」などのワープロソフトはテキストエディタではありませんので、JavaScript のプログラムを書くのには使えません。



●ブラウザは動作確認に使う

ブラウザは、Webページを見るためのソフトですが、同時に、**JavaScriptを実行するためのソフト**でもあります。ブラウザでJavaScriptのコードが書かれたHTMLファイルを開くと、JavaScriptのプログラムも一緒に実行されます。

また、HTMLファイルにJavaScriptファイルが関連付けられていれば、HTMLファイルを開いたときにプログラムが呼び出されて実行されます。

ブラウザにも、たくさんの種類があります。よく使われているブラウザは、

- ・ Internet Explorer (インターネットエクスプローラー。IEと略される)
- ・ Mozilla Firefox (モジラ・ファイアフォックス。Fxや火狐と呼ばれたりする)
- ・ Safari (サファリ)
- ・ Opera (オペラ)
- ・ Google Chrome (グーグルクロム)

などです。最新のバージョンであれば、どのブラウザであっても、JavaScriptのプログラムは問題なく実行できますが、ブラウザによって多少動作が異なる場合があります。この本では、Google Chromeを使って説明します。Google Chromeのインストール方法は、このあとのセクションで説明します。

COLUMN

その他のテキストエディタ

代表的なテキストエディタには、以下のようなソフトがあります(2014/5/28時点の情報)。それぞれ使い勝手が違い、JavaScriptのソースコードを書くために便利な機能を持つものもあります。この本ではメモ帳で説明を進めますが、お好きなソフトを使っていただいてもかまいません。

○ Windows 用テキストエディタ

◎ TeraPad (無料)

<http://www5f.biglobe.ne.jp/~t-susumu/library/tpad.html>

◎ サクラエディタ (無料)

<http://sakura-editor.sourceforge.net/>

◎ EmEditor (4,000 円～)

<http://jp.emeditor.com/>

◎ 秀丸エディタ (4,320 円)

<http://hide.maruo.co.jp/software/hidemaru.html>

○ Mac 用テキストエディタ

◎ mi (無料)

<http://www.mimikaki.net/>

◎ CotEditor (無料)

<http://coteditor.github.io/>

◎ Jedit X (2800 円～)

http://www.artman21.com/jp/jedit_x/



ブラウザを インストールする

JavaScript のプログラムを動かすためのブラウザとして、Google Chrome をダウンロードして、インストールします。

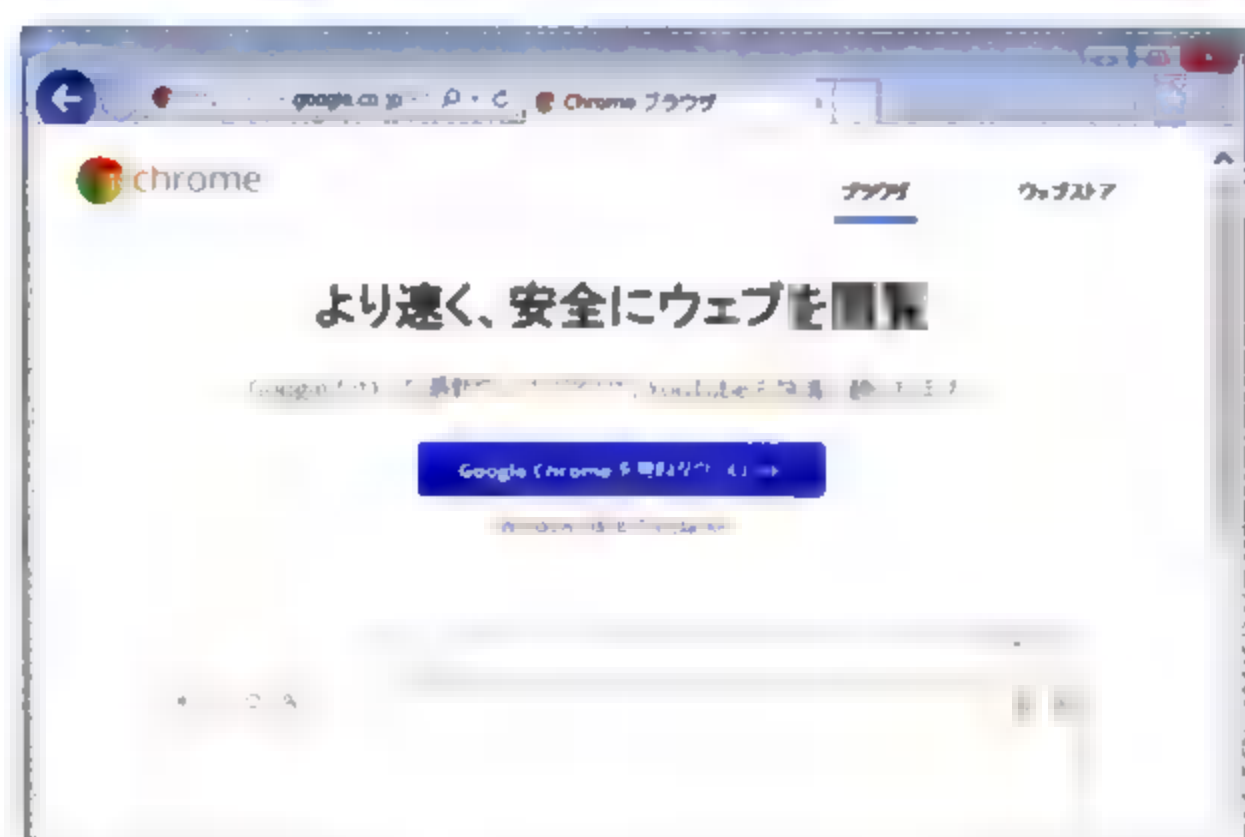
●この本では Google Chrome を使う

Windows のパソコンであれば、最初から Internet Explorer というブラウザが入っていますが、この本では、Google Chrome (以下、Chrome と略) というブラウザを使って説明していきます。Chrome は、Windows、OS X (Mac)、Android、iOS など、複数の OS で使えて世界的にシェアが大きく、また JavaScript のプログラムを作っていくうえで、いろいろと便利な機能を備えているからです。

● Chrome の入手方法

Chrome はインターネットから無料で入手できます。ふだんお使いのブラウザを起動して「Google」や「Yahoo! JAPAN」などの検索サイトを開き、「クロム」と検索してみてください。検索結果の上の方に「Chrome ブラウザ」という名前の公式サイトが見つかるかと思います。

 Google Chrome (<http://www.google.co.jp/intl/ja/chrome/browser/>)



Chrome のダウンロードページ

このサイトからダウンロードして、インストールしてください。



Lesson 1

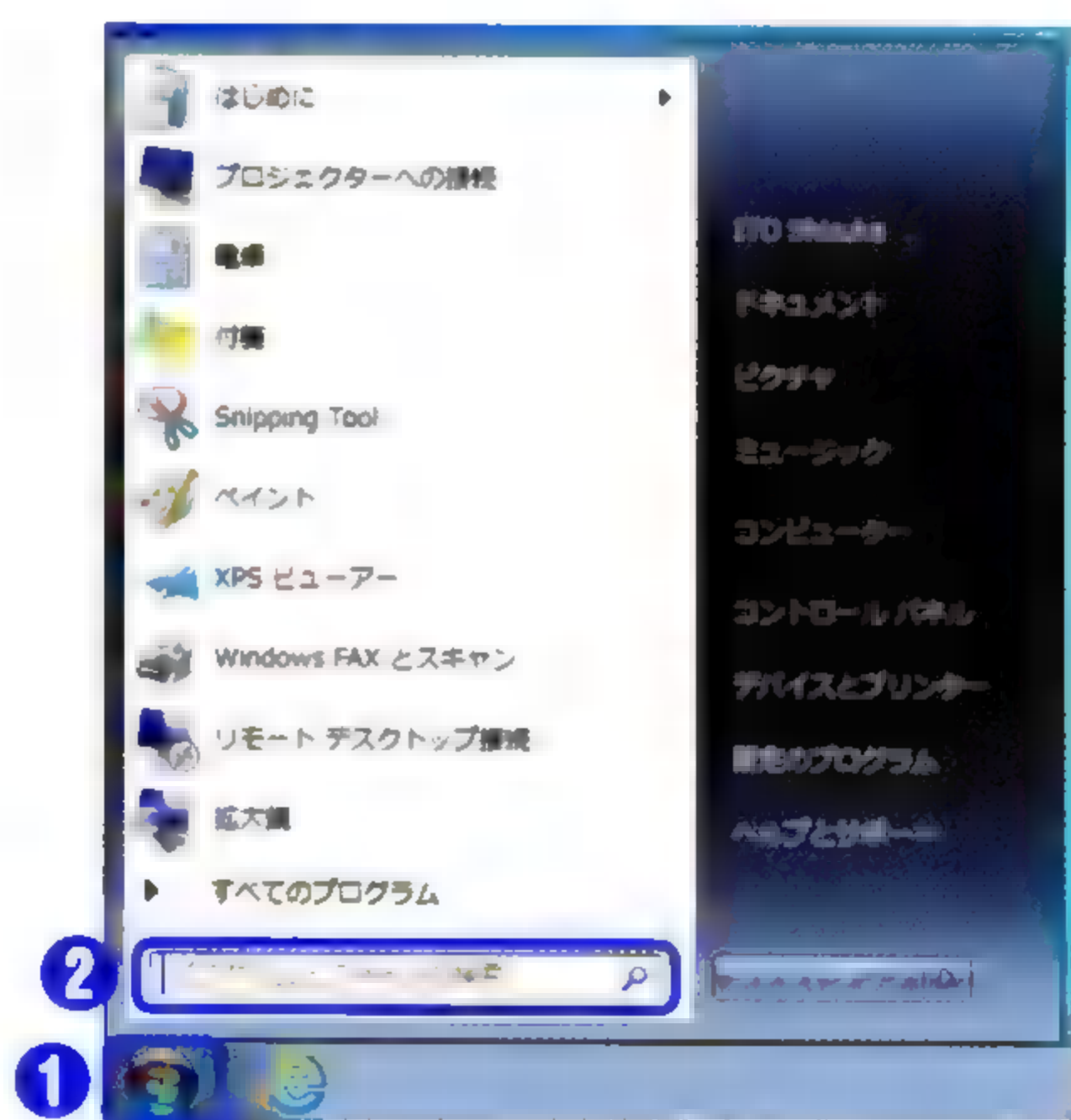


テキストエディタを 起動する

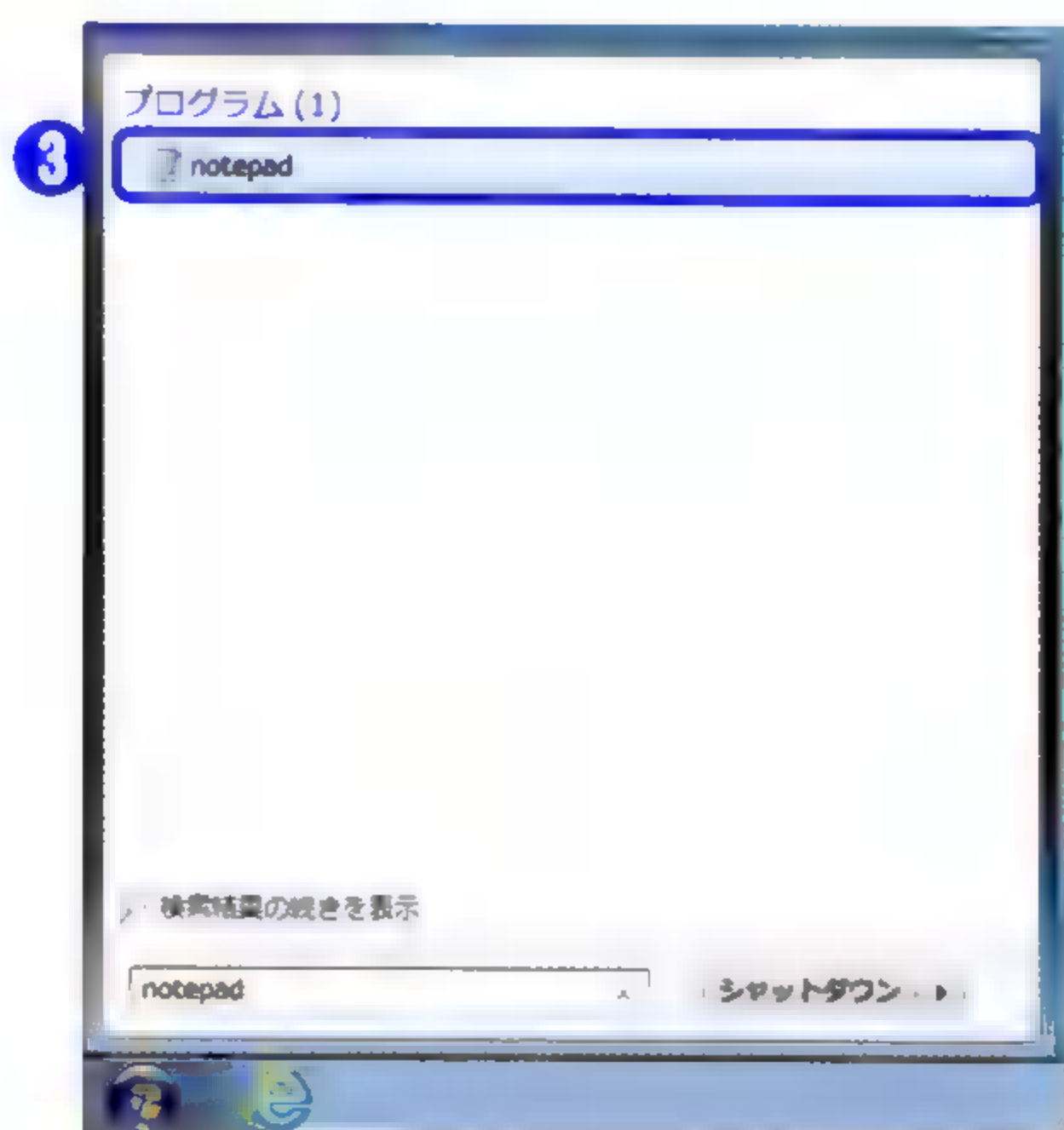
JavaScript プログラムのソースコード（中身）は、テキストエディタを使って書きます。この本では、Windows に最初から入っているメモ帳というテキストエディタを使います。

● メモ帳を起動する

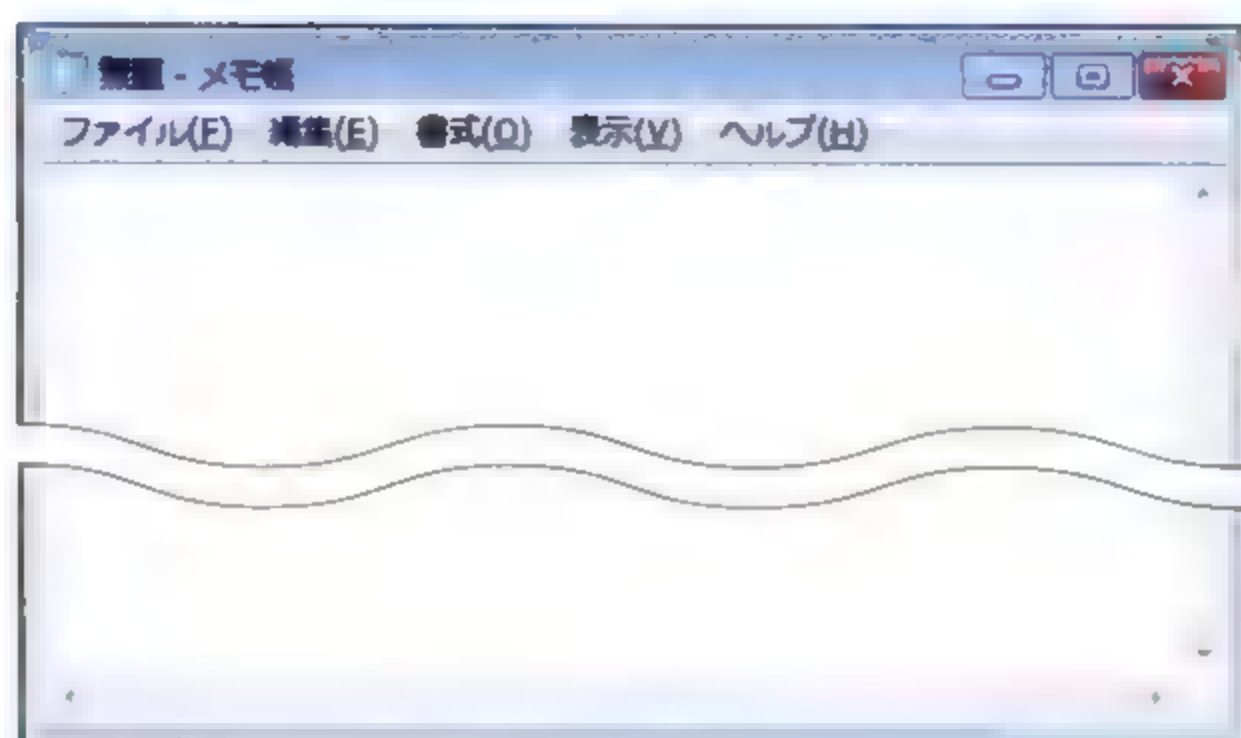
Chrome をインストールしたら、次はメモ帳を起動してみましょう。



- ① スタートボタンをクリック
- ② 検索窓に「notepad」と入力



- ③ 「notepad」をクリック



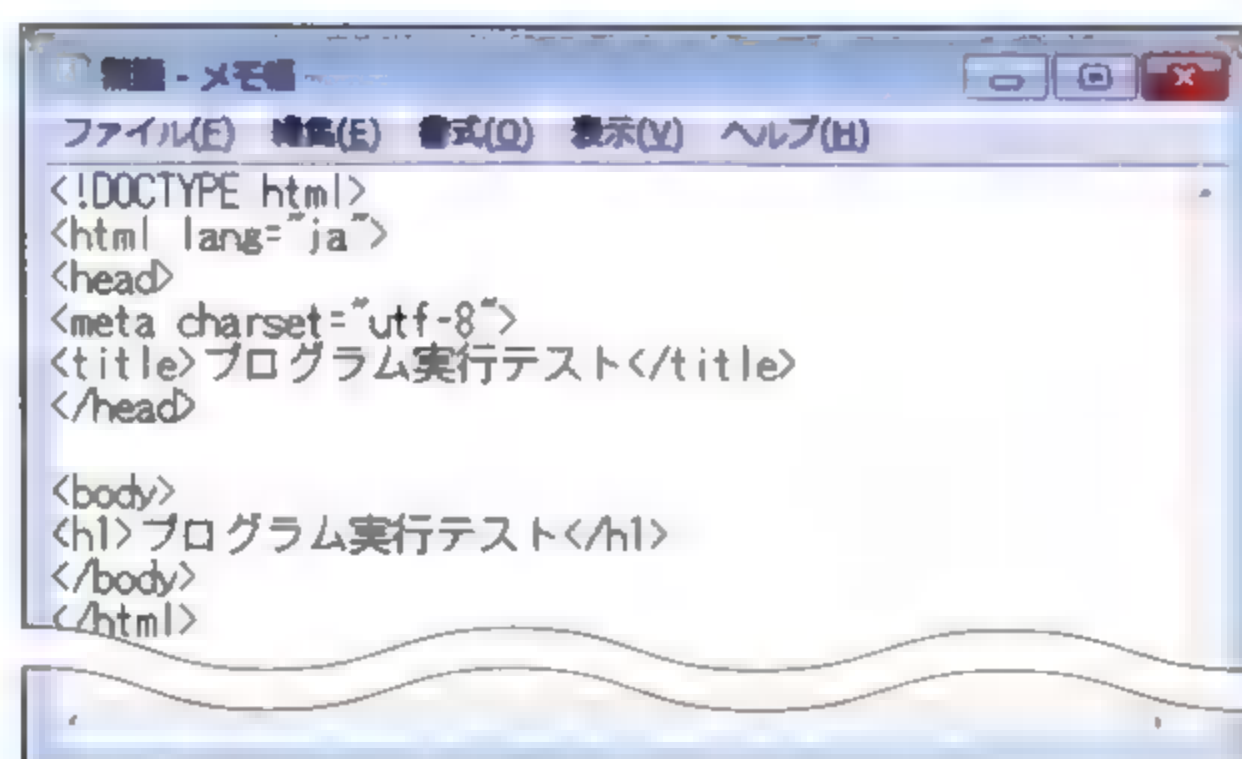
メモ帳が起動した

● HTML のソースコードを入力する

続いて、起動したメモ帳に、以下のHTMLソースコードを入力してください。

2カ所ある「プログラム実行テスト」という文字以外は、**すべて半角文字**で入力します。この本で作るサンプルプログラムでは、このHTMLソースコードを使っていきます。一度入力しておけば、このあとずっと使い回しできます。

```
1  <!DOCTYPE html>
2  <html lang="ja">
3  <head>
4  <meta charset="utf-8">
5  <title> プログラム実行テスト </title>
6  </head>
7
8  <body>
9  <h1> プログラム実行テスト </h1>
10 </body>
11 </html>
```



入力後の画面

ソースコードを入力したら、そのままの状態にしておいてください。次はこのHTMLソースコードの中に、JavaScriptのプログラムを書いてみます。

Day 1



Lesson 2

1 行だけのプログラムを作ってみる

このレッスンでは、JavaScriptの簡単なプログラムを作成してみます。このあと何度も繰り返すことになる「ソースコードを入力」→「プログラムを実行」→「プログラムを書き換え」→「再度プログラムを実行」という一連の手順を学びます。

- 1 ソースコードを書く
- 2 プログラムを実行する
- 3 プログラムを書き換えて実行する
- 4 ファイルの拡張子が見えるように設定する

Lesson 2



ソースコードを書く

JavaScript のプログラムは、HTML ファイルの script 要素の中に書きます。実際に短いソースコードを書いて、ブラウザで表示してみましょう。

●メモ帳に追加でソースコードを入力する

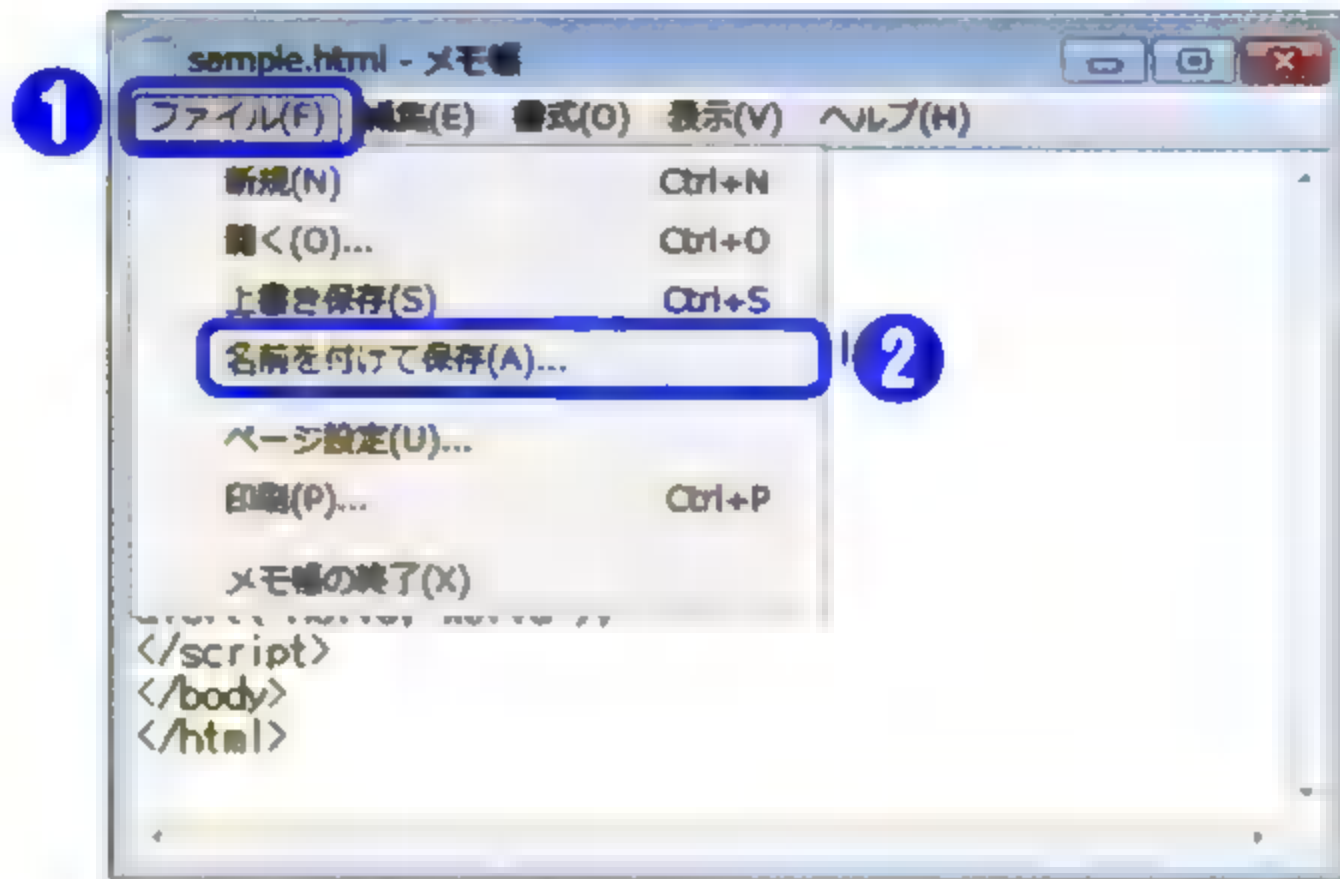
先ほど起動したままのメモ帳に、以下のソースコード（色文字部分）を追加して入力してください。

```
1  <!DOCTYPE html>
2  <html lang="ja">
3  <head>
4  <meta charset="utf-8">
5  <title> プログラム実行テスト </title>
6  </head>
7
8  <body>
9  <h1> プログラム実行テスト </h1>
10 <script>
11 window.alert('Hello, world!');
12 </script>
13 </body>
14 </html>
```

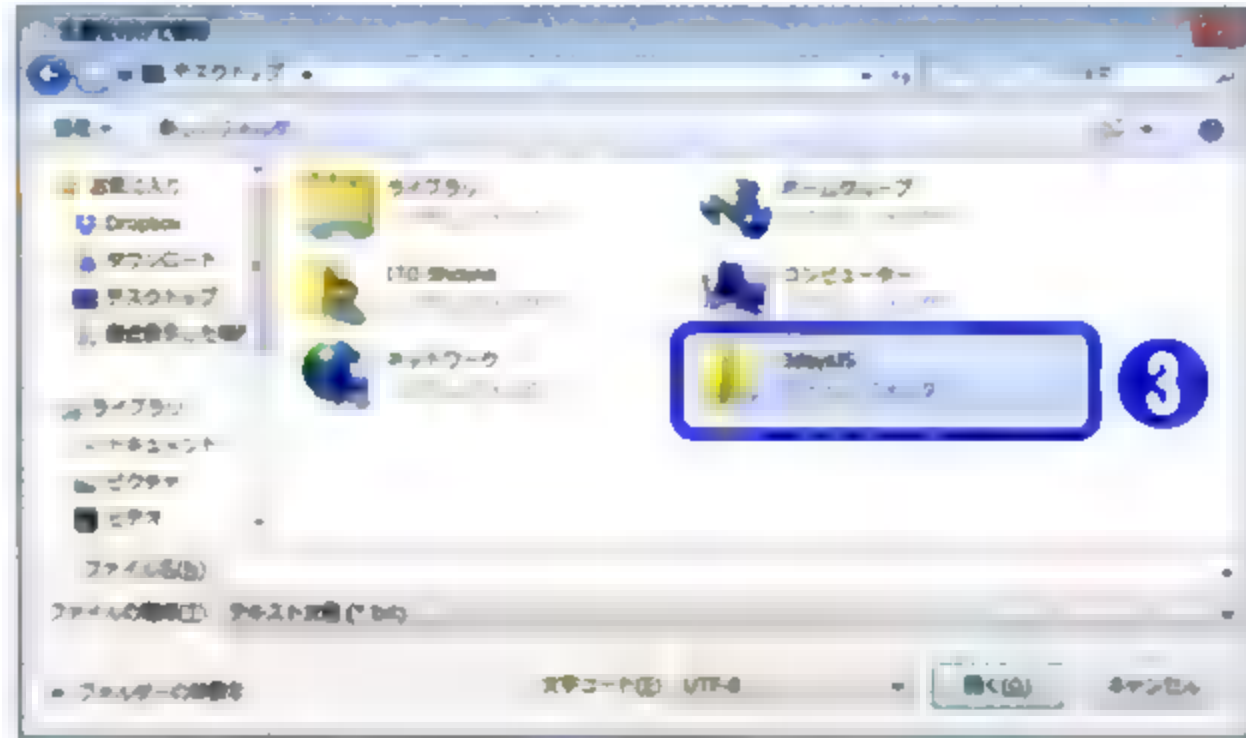
入力し終わったら、以下のよう
な手順で、HTMLファイルと
して保存します。

MEMO

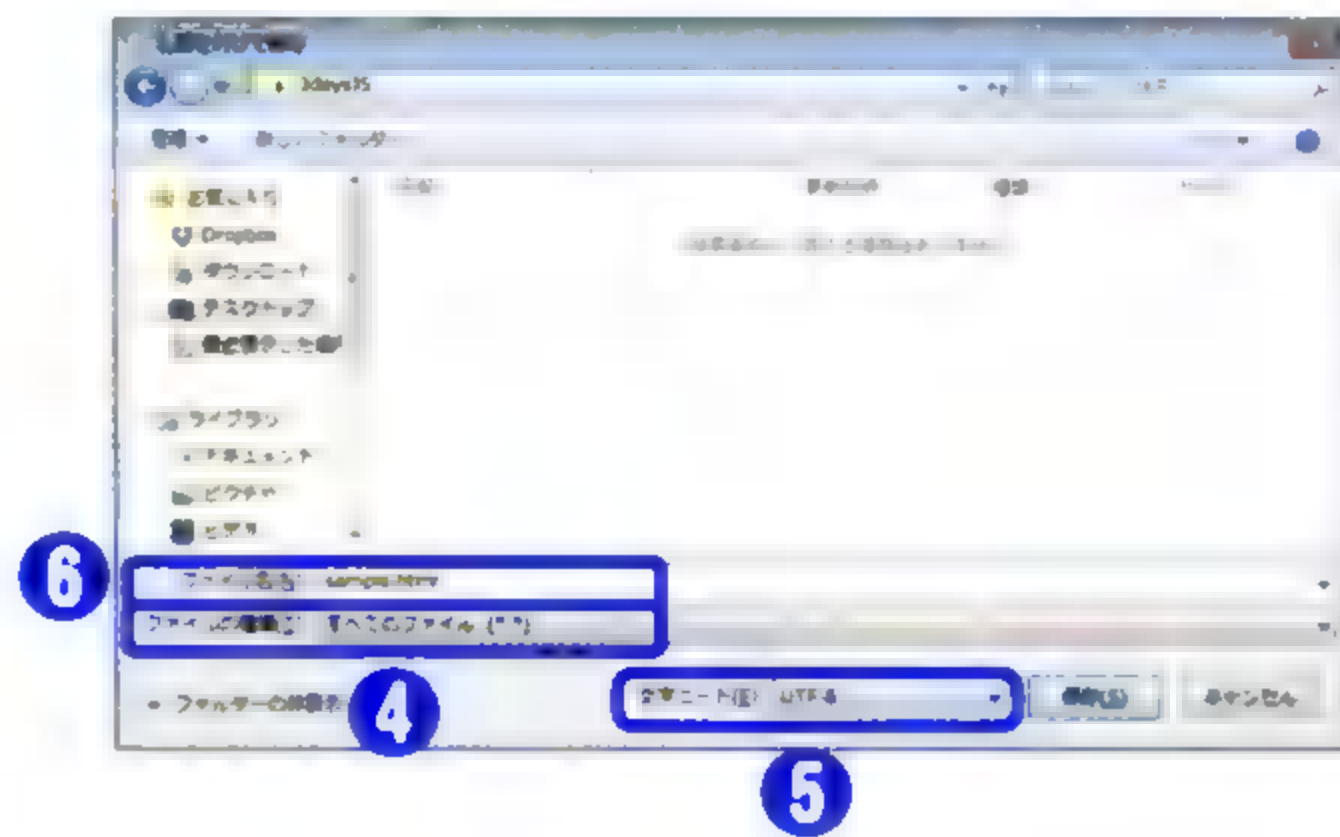
「window.alert('Hello, world!');」は警告ダイアログ
ボックスを表示する文です。文法的な意味はDay3-
Lesson1-Section2（以後、3-1-2のように略）で学
習します。



- ① 「ファイル」をクリック
- ② 「名前を付けて保存」をクリック



- ③ 保存先のフォルダをダブルクリック



- ④ ファイルの種類で「すべてのファイル」を選択
- ⑤ 文字コードで「UTF-8」を選択
- ⑥ ファイル名の欄に、半角で「sample.html」と入力して、「保存」をクリック

これでファイルの保存は完了です。次は、このHTMLファイルをブラウザで開いてみます。

MEMO

文字コードで「UTF-8」以外を選択すると、ブラウザで開いたときに文字化けします。また、この本では、HTMLファイルはHTML5の文法にのっとって書いていきます。

Lesson 2



プログラムを実行する

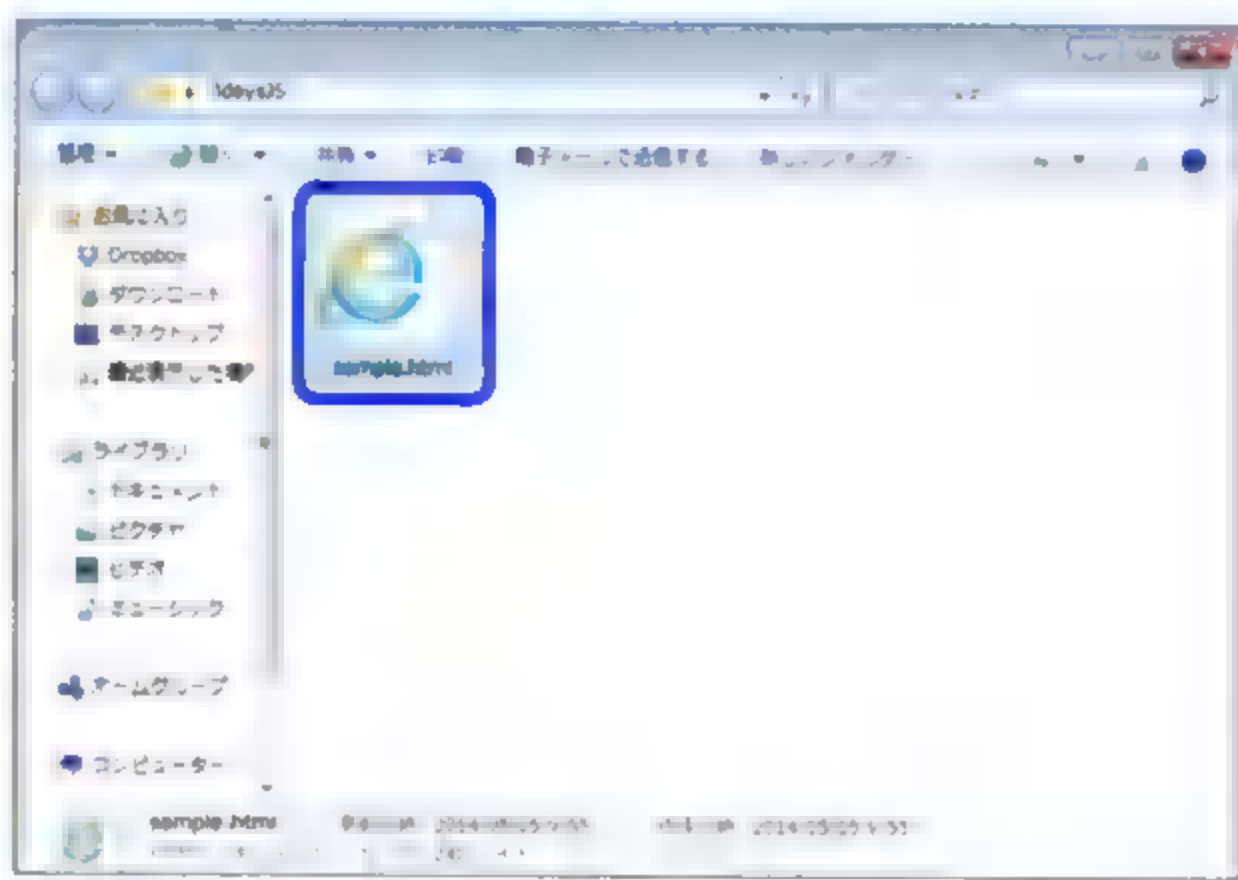
先ほど保存したHTMLファイルをブラウザで開いて、JavaScriptのプログラムを実行してみましょう。

● HTML ファイルをブラウザで開く

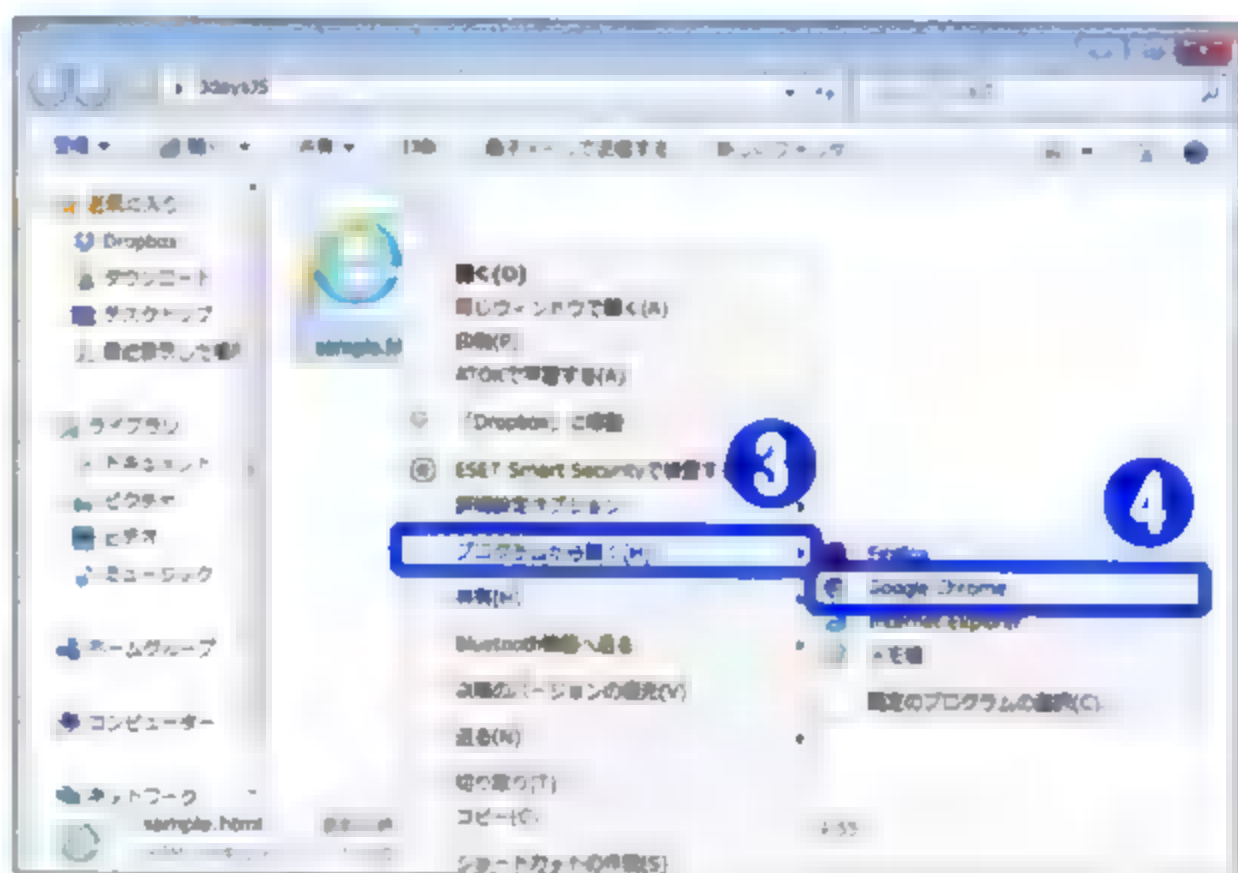
JavaScriptのプログラムを実行するには、プログラムが書き込まれたHTMLファイルをブラウザで開くだけでOKです。ブラウザがJavaScriptのプログラムを実行してくれます。sample.htmlファイルを、Chromeで開いてみましょう。



① 保存先のフォルダをダブルクリックして開く



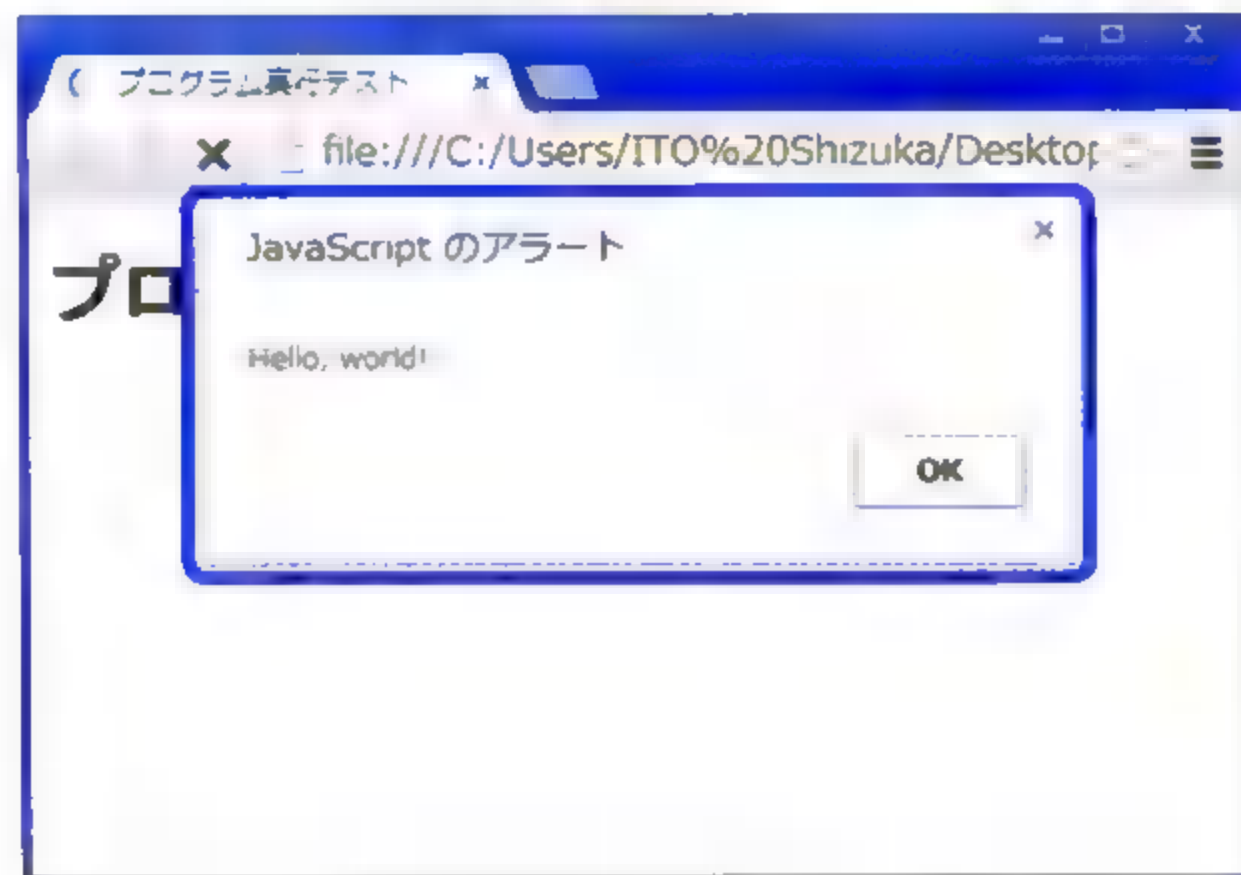
② 「sample.html」のアイコンを右クリック



③ 「プログラムから開く」にポインタを合わせる

④ 「Google Chrome」をクリック





警告ダイアログボックスが表示された

sample.htmlがChromeで開かれ、警告ダイアログボックスが表示されれば、プログラムは正しく書けています。表示されたダイアログの「OK」をクリックすれば、ダイアログボックスは消えます。

MEMO

プログラムを修正したり、加筆したりするために、HTMLファイルをメモ帳で開きたい場合は、「④「Google Chrome」をクリック」のところで「メモ帳」をクリックしてください。

MEMO

もしダイアログが表示されない場合は、HTMLのコードか、JavaScriptのコードのどこかに間違いがあるはずです。本のコードとテキストエディタに入力したコードをよく見比べてみて、修正してください。

ここまで完了すれば、これからJavaScriptのプログラムを勉強していく準備が整ったということです。

sample.htmlを表示したChromeとメモ帳はまだ閉じないでください。このまま続いて、プログラムを少し書き換えてみましょう。

JavaScript のソースコードを記入する場所

JavaScriptのコードを書く場所は、大きく分けて3カ所あります。

- ①HTML ファイルのscript要素の中
- ②a要素の開始タグの中
- ③JavaScript ファイル（拡張子.js）の中

①は先ほど使いましたので省略します。

②は、a要素のhref属性値として、URLのようにしてソースコードを書く方法です。この書き方をJavaScript疑似プロトコルといいます。ソースコードは、このようになります。

```

8  <body>
9  <h1> プログラム実行テスト </h1>
10 <a href="JavaScript:window.alert('Hello, world!');"> 警告ダイアログボッ
    クスを表示 </a>
11 </body>

```

画面に表示されたリンクをクリックすると、警告ダイアログボックスが表示されます。

③JavaScript ファイル（拡張子.js）の中に書く方法ですが、HTML ファイルとは別のファイルにJavaScriptのソースコードを書き（scriptタグは不要）、

- ・ 拡張子を.js
- ・ 文字コードをutf-8

にして名前を付けて（sample.jsなど）、たとえばHTML ファイルと同じ場所に保存します。

sample.js

```

1  window.alert('Hello, world!');

```

その上で、HTML ファイルのhead要素の中に、関連付けのためのscript要素を記入すればOKです。

```

1  <!DOCTYPE html>
2  <html lang="ja">
3  <head>
4  <meta charset="utf-8">
5  <title> プログラム実行テスト </title>
6  <script src="sample.js"></script>
7  </head>
8
9  <body>
10 <h1> プログラム実行テスト </h1>
11 </body>
12 </html>

```



Lesson 2

SECTION 3

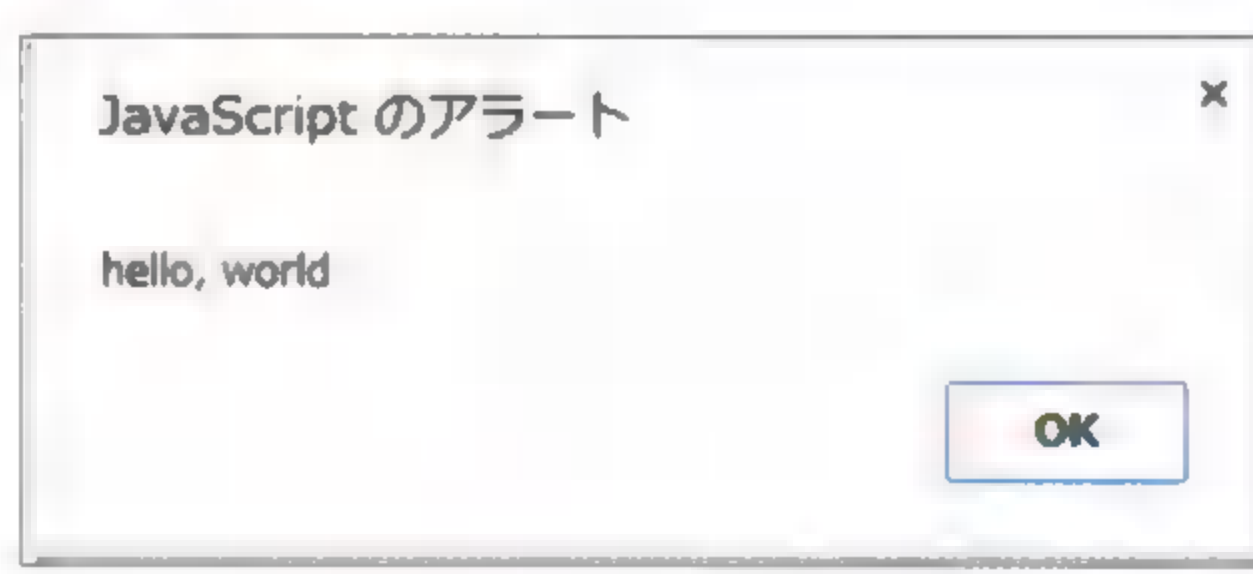
プログラムを書き換えて実行する

ブラウザで sample.html を表示したまま、テキストエディタでソースコードを書き換えてみましょう。これから先、何度もおこなう手順です。

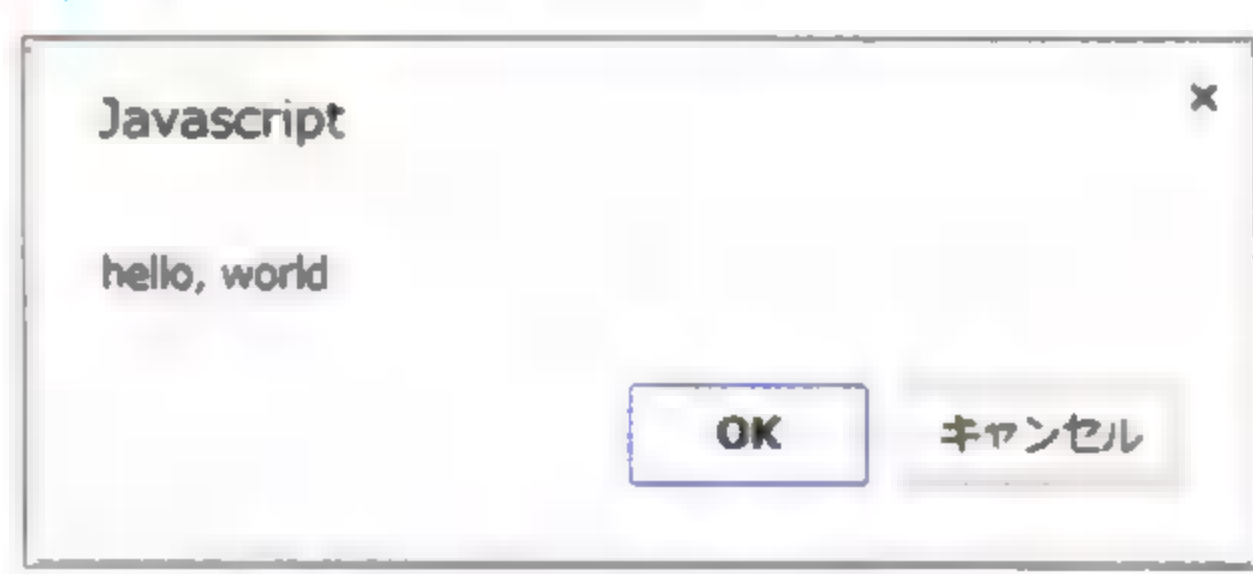
● JavaScript のソースコードを書き換える

ソースコードを書き換えて、表示されるダイアログの種類を「警告ダイアログボックス」から「確認ダイアログボックス」に変えてみましょう。

警告ダイアログボックス



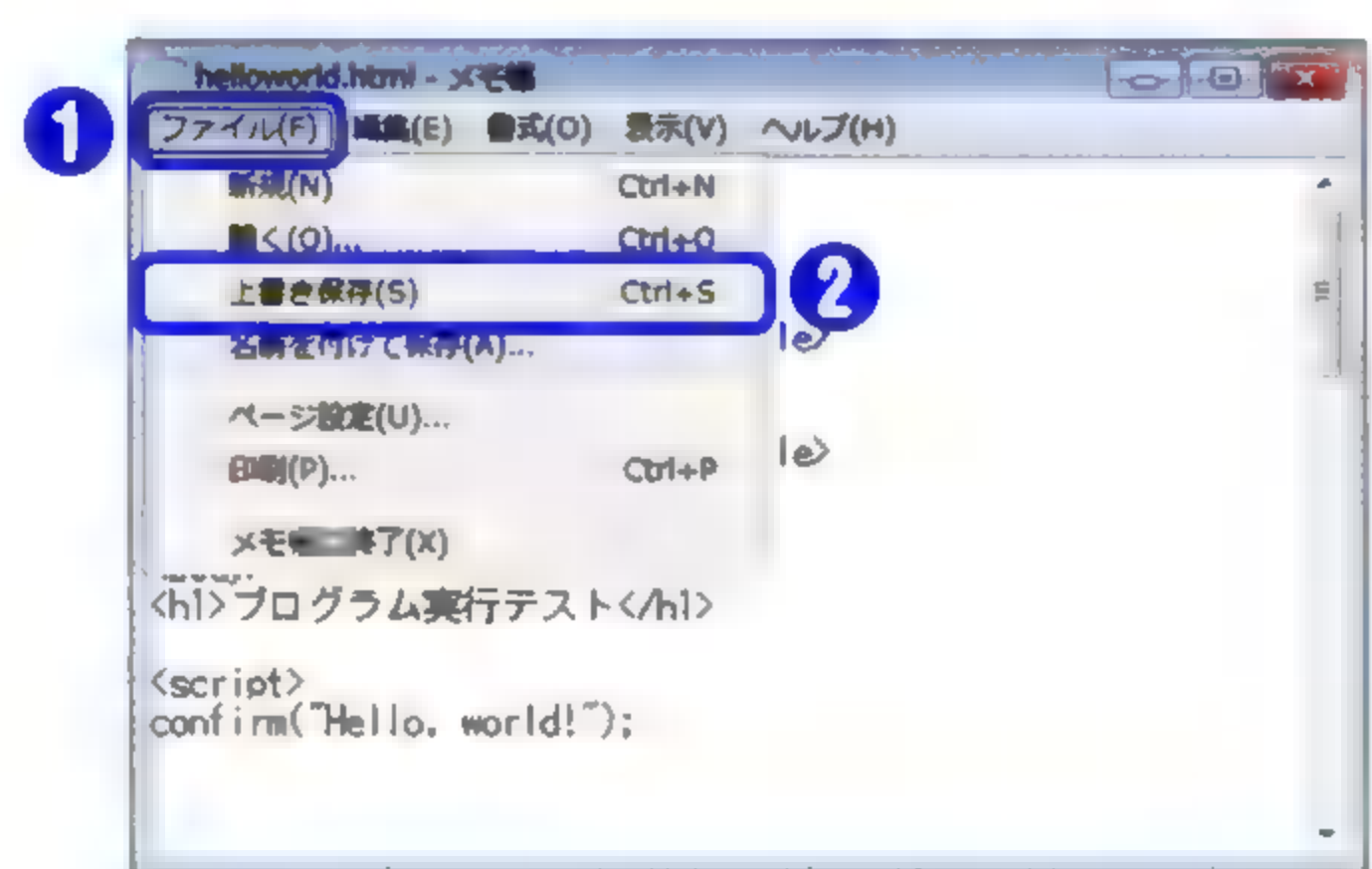
確認ダイアログボックス



メモ帳のソースコードの「alert」を、以下のように「confirm」に書き換えてください。

```
10 <script>
11 window.confirm('Hello, world!');
12 </script>
```

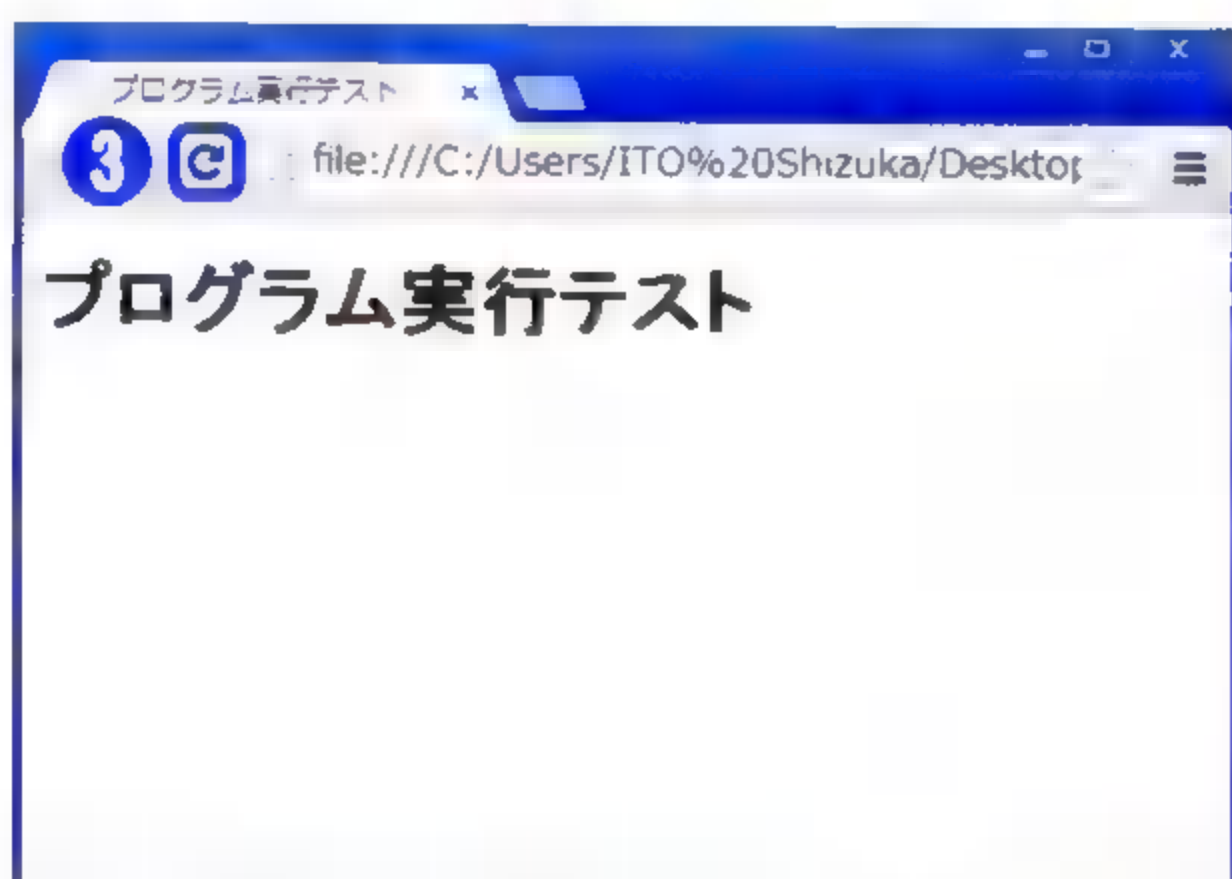
※ script 要素以外は省略しています。



- ① 「ファイル」をクリック
- ② 「上書き保存」をクリック

MEMO

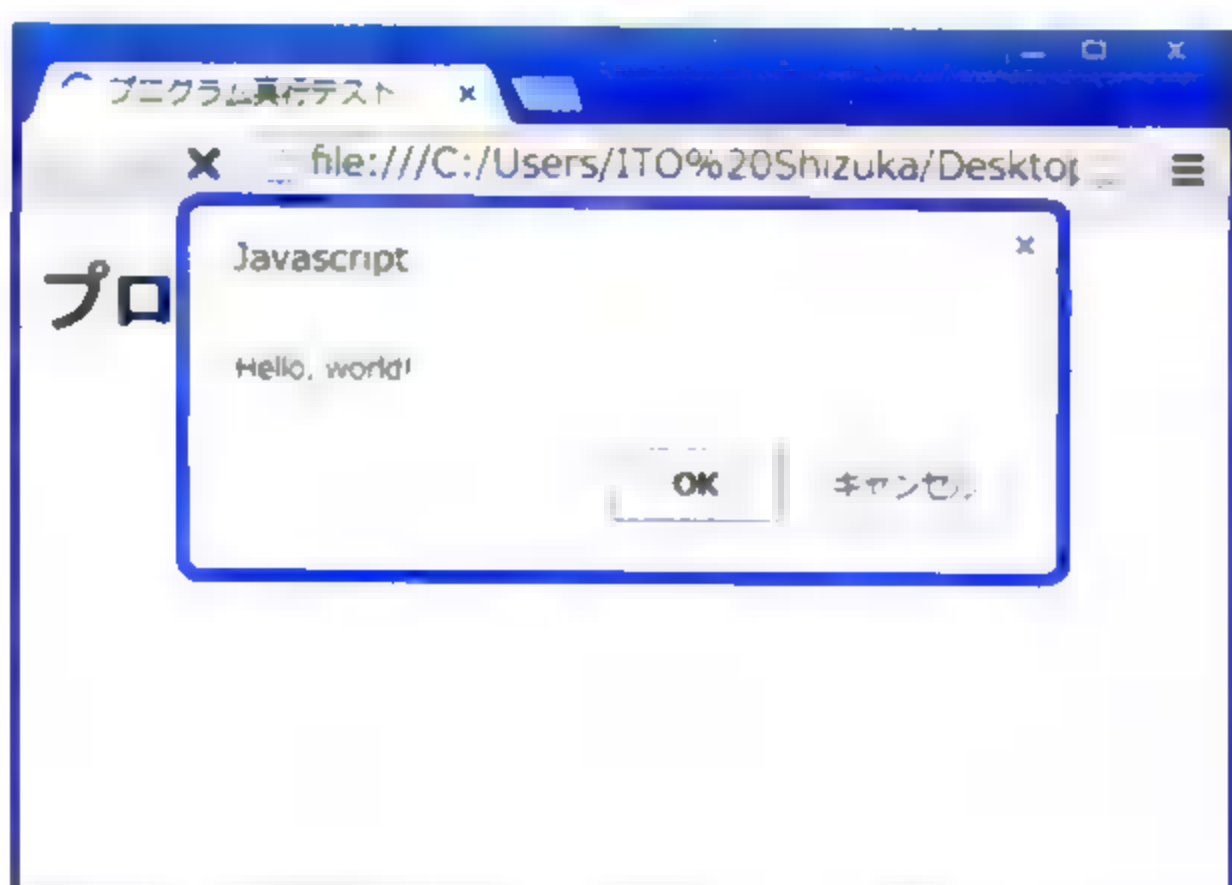
上書き保存は、Ctrl + S キーでも OK です。



③「再読み込みボタン」をクリック

MEMO

再読み込みは「リロード」ともいいます。ブラウザがアクティブになっていれば、F5キーを押しても、リロードできます。



確認ダイアログボックスが表示された

確認ダイアログボックスが表示されれば、プログラムはきちんと書き換えられています。表示されたダイアログボックスの「OK」か「キャンセル」をクリックすれば、ダイアログボックスは消えます。

今後、新しく学んだ内容を、サンプルプログラムに追加で書いたり、エラーを修正したりして実行するときには、Chromeとメモ帳で同時にsample.htmlを開いたまま、「メモ帳で上書き保存」→「Chromeで再読み込み」の手順を繰り返して、動作を確認してください。

Lesson 2



ファイルの拡張子が見えるように設定する

最後の準備として、HTML ファイルなどの拡張子が表示されるように設定しておきましょう。

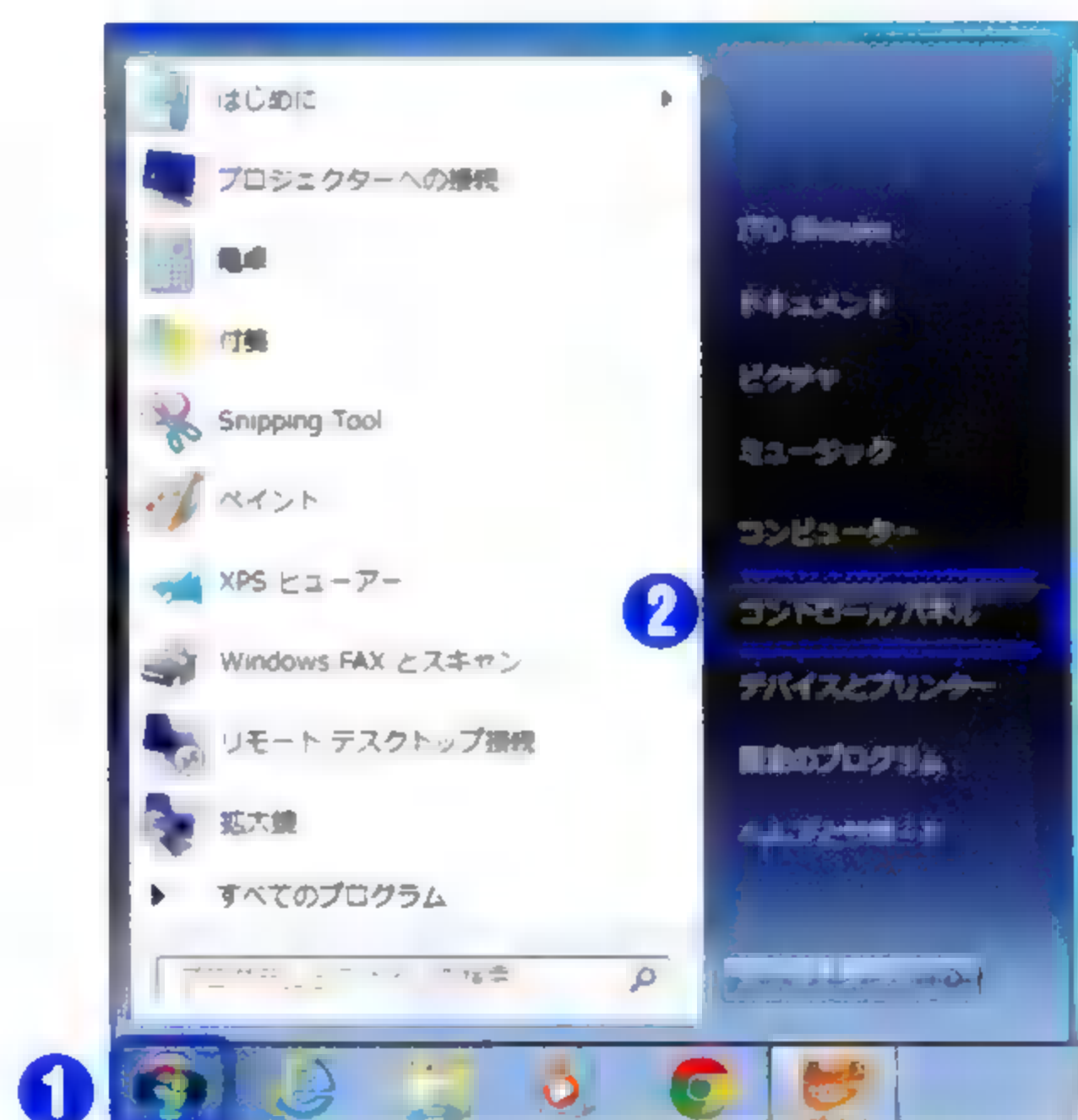
● 拡張子はファイルの種類を表す文字列

拡張子とは、ファイル名の末尾にある「ドットと英数字の文字列」のことで、ファイルの種類を表しています。HTML ファイルの拡張子は「.html」か「.htm」で、JavaScript ファイルの拡張子は、「.js」です。

Windows 7やWindows 8の初期状態では、拡張子は表示されない設定になっています。ですが、Webページを作ったり、JavaScriptのプログラムを書いたりするときには、拡張子でファイルの種類が分かった方が便利なので、拡張子が表示されるように設定を変更しておきましょう。

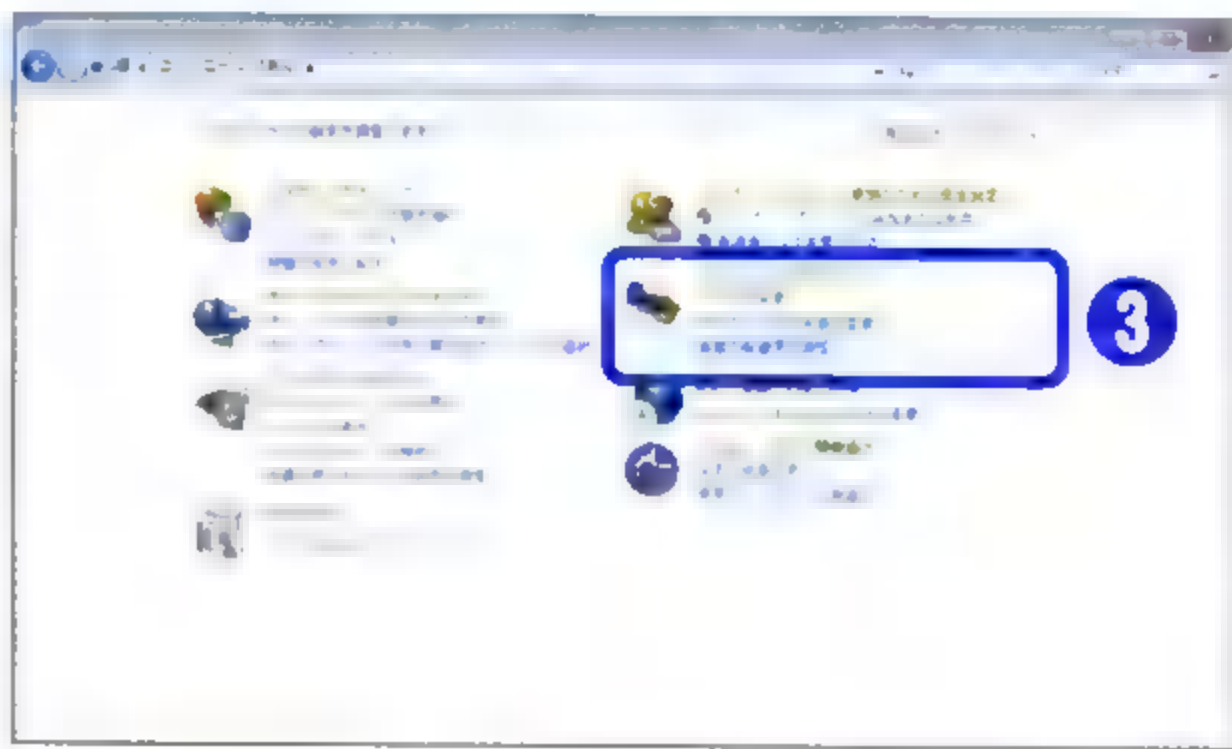
● ファイルの拡張子を表示するには？

Windows 7でファイルの拡張子を表示するには、以下の操作手順を行ってください。

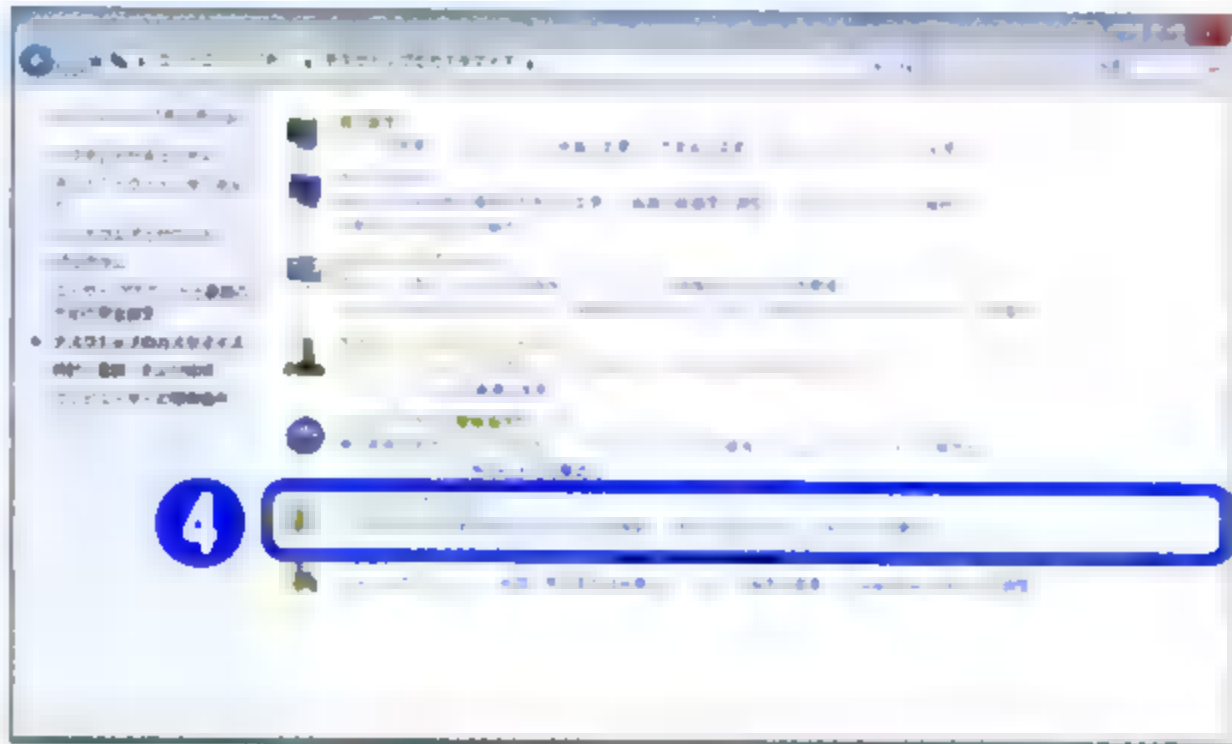


① スタートボタンをクリック

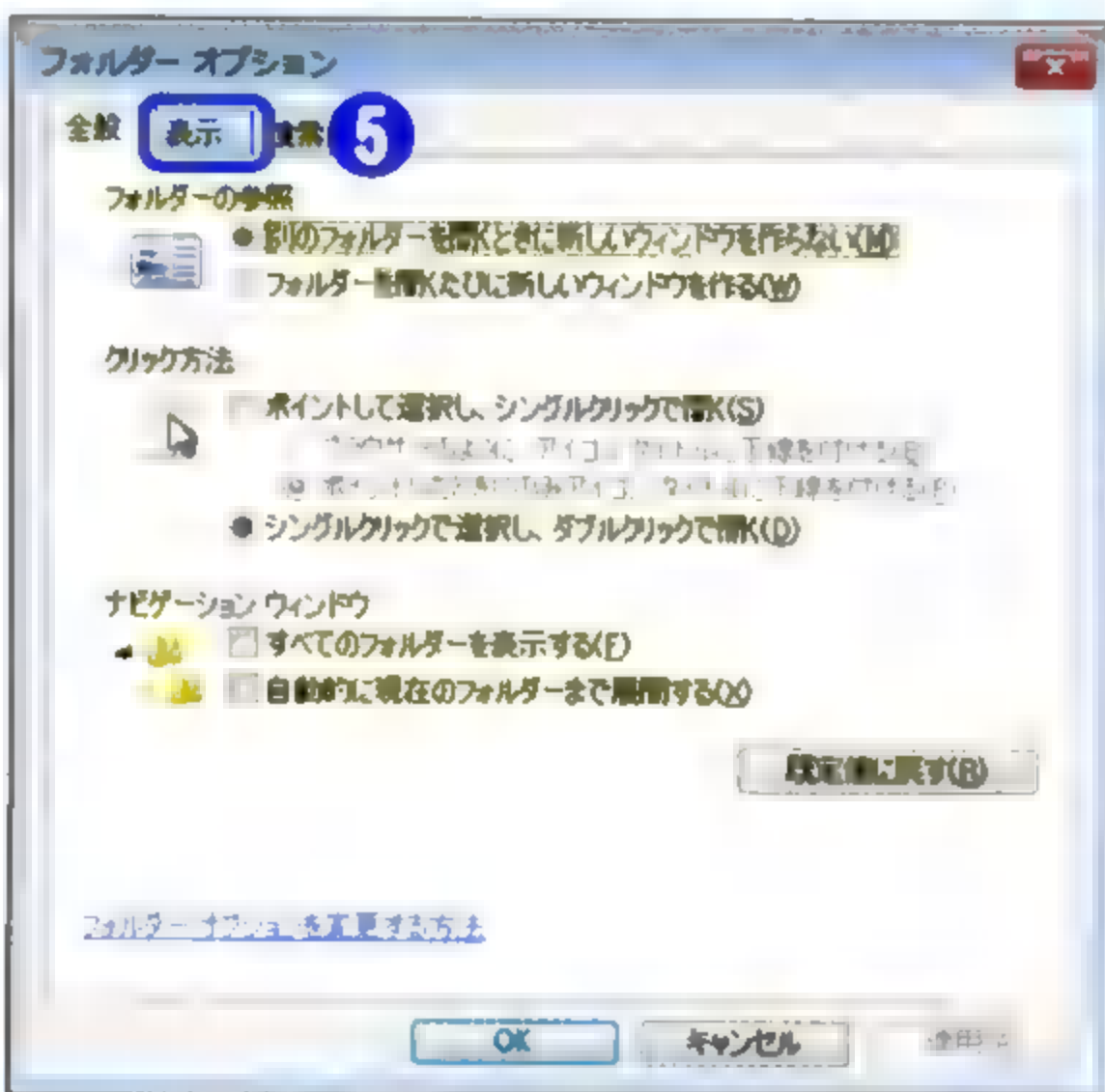
② 「コントロールパネル」をクリック



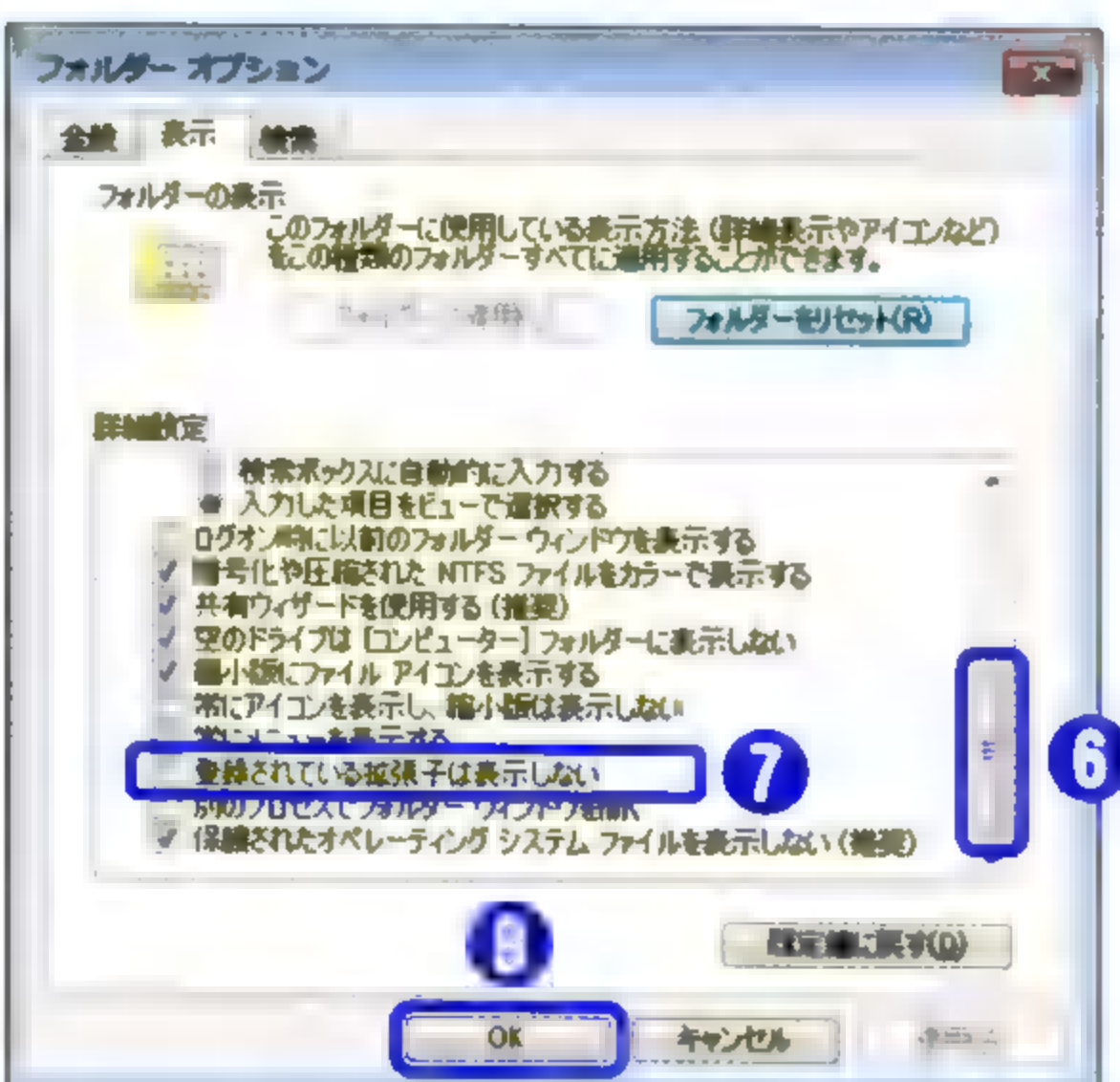
③ 「デスクトップのカスタマイズ」をクリック



④ 「フォルダーオプション」をクリック



⑤ 「表示」をクリック



⑥ スライダーを下までドラッグ

⑦ 「登録されている拡張子は表示しない」のチェックを外す

⑧ 「OK」をクリック

これで拡張子が表示される設定になりました。





COLUMN

Windows 8 で拡張子を表示するには？

- ① キーボードの最下段にある「Windows マーク」キーを押しながら X キーを押す
- ② 表示された一覧から「コントロールパネル」をクリックして、コントロールパネルを開く
- ③ 「デスクトップのカスタマイズ」をクリック
- ④ 「フォルダーオプション」をクリック
- ⑤ 「表示」をクリック
- ⑥ 「詳細設定」ボックスのスライダーを下までドラッグ
- ⑦ 「登録されている拡張子は表示しない」のチェックを外す
- ⑧ 「OK」をクリック

これで拡張子が表示される設定になります。

COLUMN

HTML ファイルの拡張子をブラウザに関連付ける

HTML ファイルの拡張子である「.html」と「.htm」を Chrome に関連付けておけば、HTML ファイルのアイコンをダブルクリックするだけで、自動的にそのファイルが Chrome で開くようになります。JavaScript の学習を進めていくうえでは、何度も HTML ファイルを開くこととなりますので、面倒な手間は少しでも省けるようにしておきましょう。

- ① 「スタート」メニューの「既定のプログラム」をクリック
- ② 「ファイルの種類またはプロトコルのプログラムの関連付け」をクリック
- ③ 「ファイルの種類またはプロトコルを特定のプログラムへ関連付けます」が開くので、拡張子の一覧から「.html」を見つけて、クリックして選択
- ④ 「プログラムの変更」ボタンをクリック
- ⑤ 「Google Chrome」を選択して「OK」ボタンをクリック

同じようにして、念のため、拡張子「.htm」も Chrome に関連付けておきましょう。また、拡張子「.js」はメモ帳など、使っているテキストエディタに関連付けておきましょう。



Day 1



Lesson 3

プログラム作成前に 必要な知識

このレッスンでは、そもそもプログラムとは何なのかを簡単に説明したうえで、ソースコードを書くときの注意事項を説明します。

- 1 プログラムとは何か？
- 2 ソースコードの書き方

プログラムとは何か？

今回初めてプログラミングを学ぶという人向けに、プログラムの基本的な構造を説明しておきます。この本でこれからどういうことを学ぶのか、大まかにつかんでください。

●プログラムとは？

一般にコンピュータのプログラムとは、「コンピュータに実行してほしい内容を書いた指示書」です。これをJavaScriptというプログラミング言語で書くとJavaScriptのプログラムになりますし、PHPという言語で書くとPHPのプログラムになります。

コンピュータとは何なのかをひとことで言えば、「データ」を「処理」する機械です。コンピュータに何かをしてもらうには、してほしいことを「データ」と「処理」に分解してプログラムに書き、コンピュータに伝えてあげないといけません。ですので、本書で学ぶ内容は、大きく分けると、

①データ

②データを処理する手順

の2つになります。

MEMO

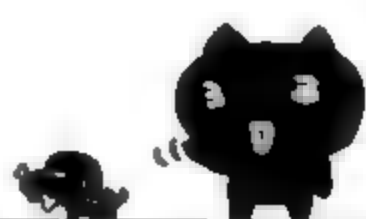
処理手順のことを、プログラミング用語で「アルゴリズム」といいます。

●データの学習内容

プログラムで使うデータは、「変数」と呼ばれる箱のような入れ物の中に保存して使います。よって「データ」の学習は、おもに「データの種類」と「変数の使い方」についての学習になります。

MEMO

この本で、変数の使い方として学ぶ内容は、データ型、文字列、数値、論理値、配列、オブジェクトなどです。



● 処理の単位＝文

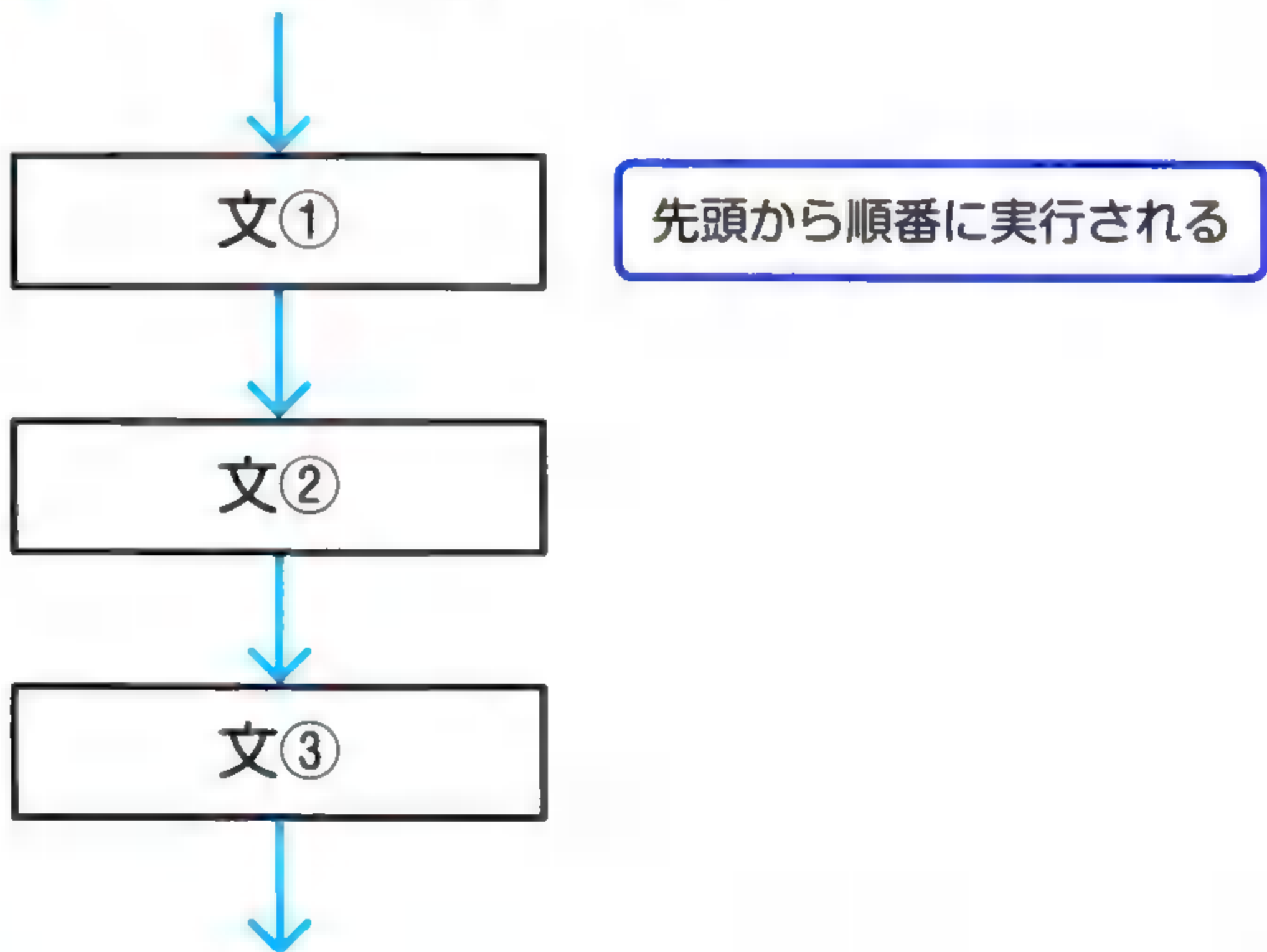
次に、プログラムのもう1つの要素である「処理手順」について説明しておきます。プログラムの中に書く1つ1つの処理は、**文**という単位で記述します。先ほど作ったプログラムは、文が1つだけのプログラムです。

```
<script>
window.alert('Hello, world!');
</script>
```

● 処理手順①：先頭から順番に実行する

複数の文が書かれたプログラムは、先頭の文から順番に実行されます。これがプログラムのもっとも基本的な処理手順です。

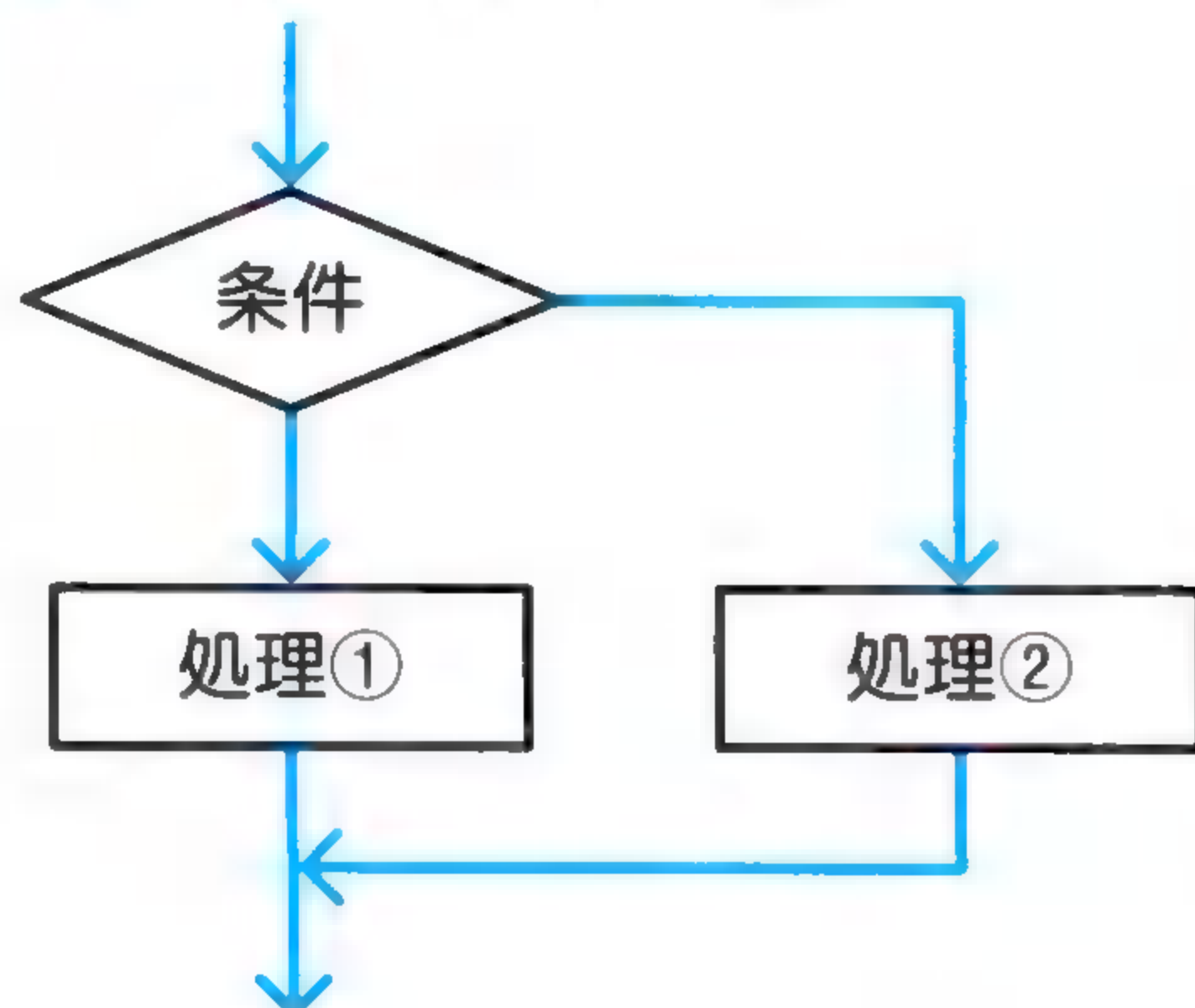
▼ 処理手順①：先頭から順番（順次）



● 処理手順②：条件で分岐して実行する

もし、処理の流れを変えたいときには、**制御文**という種類の文を使います。たとえば、何らかの条件によって処理手順の順番を変えたいときは、制御文の1つである**条件文**を使って処理を分岐させます。これが2番目の処理手順である**条件分岐（選択）**です。

▼ 処理手順②：条件分岐（選択）



条件によって処理の流れを変える

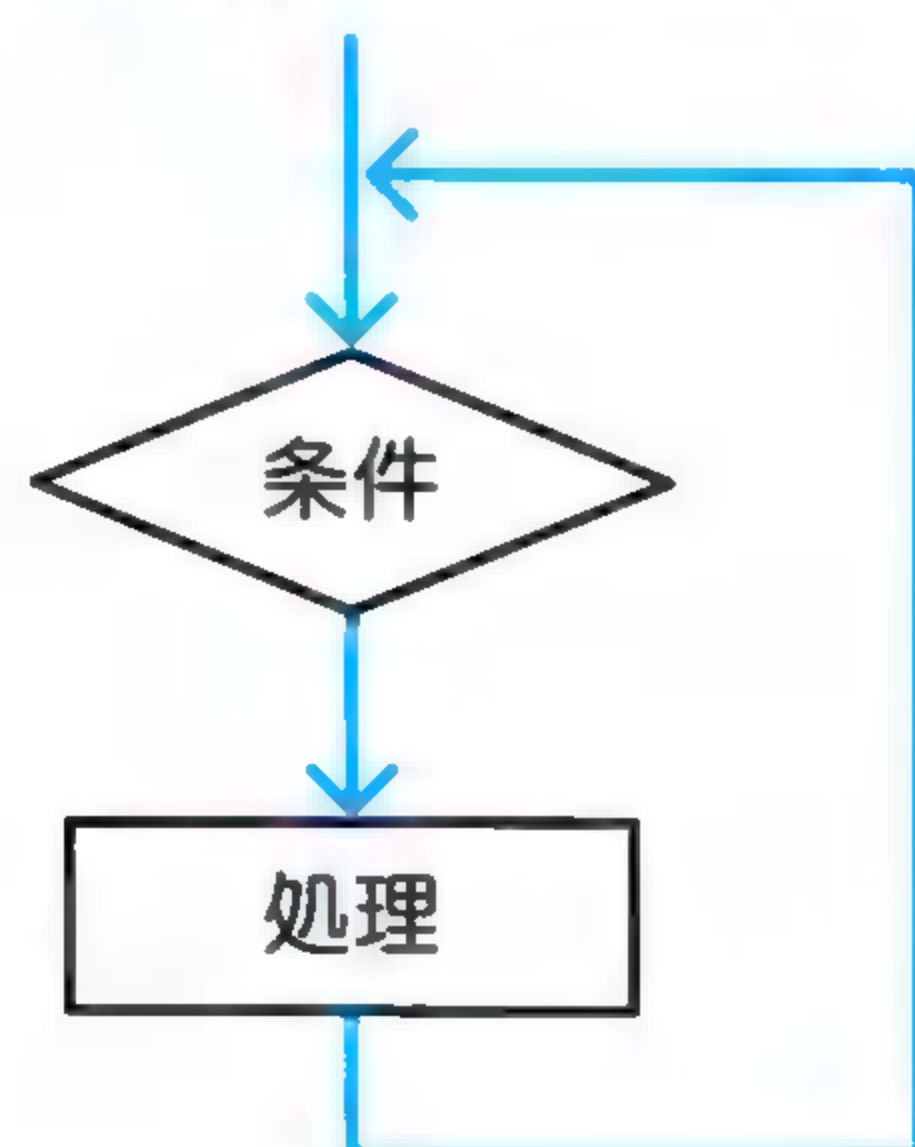
MEMO

条件文には、if文、if～else文、if～else if～else文、switch文などがあります。

● 処理手順③：繰り返し実行する

また、一定の条件のあいだ、同じ処理を繰り返し実行したいときには、おなじく制御文の1つである**繰り返し文（反復文）**を使います。

▼ 処理手順③：繰り返し（反復）



条件が満たされているあいだ
同じ処理を繰り返す

MEMO

繰り返し文には、while文、for文、do while文などがあります。

以上の内容は、だいたいどのプログラミング言語でも共通している内容です。JavaScriptを学ぶということは、これらを「JavaScriptというプログラミング言語でどのように書くか？」を学ぶということになります。





ソースコードの書き方



これからいろいろな文法を学んでいきますが、まずはソースコードを書くときに知っておかなくてはならない最低限の決まりについて説明します。

● 半角文字で書く

JavaScriptのプログラムの中身（ソースコード）は、すべて半角文字で書いてください。半角文字とは、

- ・ 半角アルファベット
- ・ 半角数字
- ・ 半角記号
- ・ 半角スペース（タブもOK）

MEMO

全角スペースは、プログラムではスペースと見なされないで、ソースコードを書くのに使ってははいけません。全角スペースを使っているとエラーになります。

です。半角カタカナは確かに半角ですが、ソースコードを書くのに使ってははいけません。

タブは厳密には半角文字ではありませんが、ソースコードを書くのに使えます。タブや半角スペースは、いくつ書いてあっても、余分なものは無視されますので、ソースコードの見た目を見やすく整えるのに使ってください。

基本は半角文字ですが、シングルクォート、ダブルクォートの中だけは、全角文字が使えます（これについては、1-4-3で説明します）。

● 大文字と小文字の違いに注意する

JavaScriptの文法では、アルファベットの大文字と小文字は別の文字として扱われます。たとえば、先ほどのサンプルプログラムで使った「alert」ですが、これを「Alert」と書いたり、「ALERT」と書いたりするとエラーになり、プログラムが動きません。ソースコードを書くときは、大文字と小文字の違いにはくれぐれも注意して書きましょう。

- alert (すべて小文字)
- × Alert (先頭が大文字)
- × ALERT (すべて大文字)

● 文の終わりは「セミコロン (;)」と「改行」

前のセクションで、プログラムの中に書く1つ1つの処理は、**文**という単位で記述すると説明しました。文と文のあいだは、**改行**か**セミコロン (;)**を使って区切ります。両方を同時に使ってもOKです。

▼ 文と文の区切り方

文① ; 文② ; 文③ ; 文④

セミコロン (;) だけ

文①

文②

改行だけ

文① ;

文② ;

改行とセミコロン

改行だけ、あるいはセミコロン (;) だけでも、文の終わりから見なされますので、わざわざ両方を使う必要はないのでは、と思うかもしれません。

しかし、JavaScriptの文法には、「1行が長い文の場合、途中で改行しても良い」という規則もあるのです。そのため、文の終わりを改行だけにしていると、ソースコードをあとから見直したり修正したりするとき、その改行が文の途中なのか、文の終わりなのかを、つねに注意しなくてはなりません。ソースコードが複雑になってくると、文の一部をうっかり消し忘れてエラーになることがあります。

プログラマとしては、エラーの原因となりうることは、できるだけ避けるようにした方が無難です。そういうわけで、1つの文の終わりは、「セミコロン (;) + 改行」とする習慣を付けておきましょう。

● ソースコードは読みやすく書く

ソースコードが複雑になってくると、だんだん読みにくくなってきます。読みにくいソースコードはエラーの元です。あとで改良するときにも時間がかかります。

ソースコードを書くときには、必要に応じて、半角スペースやタブによる**インデント (行頭字下げ)**や**コメント**などを使ってください(コメントは1-4-7で説明します)。

Day 1



Lesson 4

変数のしくみと 単純なデータ

このレッスンでは、基本的な変数の使い方
と3種類の単純なデータを学びます。

- 1 変数の使い方
- 2 データの種類 1 —— 数値
- 3 データの種類 2 —— 文字列
- 4 データの種類 3 —— 論理値
- 5 算術演算子
- 6 結合演算子
- 7 コメントの書き方
- 8 データ型



変数の使い方

プログラムで使うデータを保存しておく入れ物、変数について学びます。

●変数とは？

変数とは、プログラムで処理したいデータを保存しておくための入れ物で、イメージとしては箱のようなものです。

変数を使いたいときは、その都度、プログラマが自分で新しい変数を作らなくてはなりません。新しく変数を作ることを、**変数を宣言（定義）する**といいます。



変数はデータを入れる箱

●変数の宣言のしかた

変数を宣言するのは簡単です。「変数の名前」を考えて、次のように宣言文を書きます。

▼ 構文：変数の宣言文

```
var 変数名 ;
```

MEMO

「var」は、英語で「変数」を意味する単語である「variable」の、先頭から3文字を取って略したものです。

●変数名の付け方のルール

変数の名前、すなわち変数名は、プログラマが自由に付けてかまいませんが、次のようなルールがあります。

- ・変数名に使える文字は、**半角アルファベット**、**半角数字**、**半角アンダースコア** (`_`)、**半角ドル記号** (`$`)
- ・半角数字は先頭の文字には使えない
- ・アルファベットの大文字と小文字は区別される
- ・**予約語** (p.40) は使えない

変数名を付けるときは、中に入っているデータがどういうデータなのかがすぐに分かる名前にしておくとう便利。たとえば、西暦を表す数値データを入れておく変数名であれば「year」にしておくといった具合です。

●変数にデータを代入するには？

宣言した変数にデータを入れることを、**データを代入**するといいます。たとえば、「year」という名前の変数を宣言してから「2014」という数値データを代入するには、このように書きます。

```
var year;  
year = 2014;
```

イコール記号(=)は、**変数(左辺)に、データ(右辺)を代入する**という処理を表す記号で、プログラミング用語で**代入演算子**といいます。左辺と右辺を逆に書いてはいけません。変数の宣言と代入は、次のように1行でまとめて書くこともできます。

```
var year = 2014;
```

MEMO

宣言した変数に初めてデータを代入することを、プログラミング用語で**初期化**といいます。イコールの左右に半角スペースを入れているのは、見やすくするためです。文法上はなくてもかまいません。

●変数を参照するには？

変数に代入したデータを見たり、他の文で使ったりすることを**変数を参照する**といいます。変数を参照するには、文の中に変数名を書くだけでOKです。

たとえば、変数aに入っている数値データを、新たに宣言した変数bに代入したいときには、右辺に「a」と書きます。

```
var a = 2;  
var b = a; // 変数 b に変数 a の値 (2) が代入された
```

MEMO

「//変数bに変数aの値 (2) が代入された」はコメントです。1-4-7で説明します。

COLUMN

予約語

予約語というのは、JavaScriptの構文で使われる単語なので、あらかじめJavaScriptが「予約」している単語のことです。これらは変数名には使えません。

JavaScript の予約語

break	case	catch	continue	debugger	default
delete	do	else	finally	for	function
if	in	instanceof	new	return	switch
this	throw	try	typeof	var	void

この他に、将来、予約語になる可能性のあるキーワードというものもあります。現時点で、エラーになるものとならないものがあります。これらも変数名には使わない方がいいでしょう。

将来予約語になる可能性のあるキーワード

class	enum	export	extends	import
super	implements	interface	let	package
private	protected	public	static	yield

変数名に予約語を使ったプログラムを実行すると、エラーになり、ブラウザの画面に何も表示されません。特にエラーを示す表示はありませんので、プログラムを実行したのにブラウザで何も起こらないときは、変数名に予約語を使っていないか確認してみましょう。



Lesson 4



データの種類①

——数値

変数に代入できるデータの種類の1つずつ順番に学んでいきましょう。1つ目のデータは、数値です。数値はいろいろな計算に使えます。

●数値とは？

JavaScriptでの数値データとは、整数と小数を指します。整数は、負の整数ももちろんOKです。分数や無理数などは扱えません。

●変数に数値を代入するには？

変数に数値を代入する方法は、すでに前のセクションで紹介しました。変数に数値を代入するには、

▼ 構文：数値の代入文

変数名 = 数値 ;

と書きます。たとえば、yearという変数に「2014」という整数を代入したり、averageという変数に「-0.23」という小数を代入したりするには、このように書きます。

```
var year = 2014;  
var average = -0.23;
```

数値は、1-4-5で紹介する算術演算子を使って、計算をすることができます。

MEMO

ソースコードの文中にダイレクトに書かれた数字のことを、プログラミング用語で**数値リテラル**と呼びます。同じ数字でも、シングルクォートやダブルクォートで囲むと、このあと説明する文字列リテラルとなります。違いに注意してください。

データの種類②

——文字列



変数に代入できるデータの種類、2つ目は文字列です。文字列というのは、一般に「文字」や「テキスト」と言われる文字データの、プログラミング的な言い方です。

●文字列とは？

文字データのことを、プログラミング用語で「文字列」といいます。

- ・ アルファベットの大文字・小文字（全角、半角ともに）
- ・ 日本語のひらがな、カタカナ、漢字
- ・ 数字、記号、スペース（全角、半角ともに）

などはすべて文字列です。文字列を変数に代入するとき、ソースコードの中では、半角のシングルクォート (') か、ダブルクォート (") で囲みます。囲んだひとかたまりが、それぞれ1つの文字列データと見なされます。

▼いろいろな文字列

'Hello, world!'

"ニューヨークは、今日もいい天気です。"

"03-5217-2420"

それぞれが
1つの文字列データ

MEMO

ソースコードの中で、シングルクォート (') やダブルクォート (") で囲まれた文字列のことを、プログラミング用語で**文字列リテラル**と呼びます。

文字列リテラルを囲むはじめのクォートと終わりのクォートは、シングルのならシングル、ダブルならダブルといったように同じ種類を使う必要があります。

●変数に文字列を代入するには？

1つの変数に1つの文字列を代入するには、**代入演算子(=)**のあとに、代入したい文字列を書きます。

▼構文：文字列の代入文

変数名 = '文字列';

たとえば、「familyName」という変数に「田中」という文字列を代入するなら、このように書きます。

```
var familyName = '田中';
```

MEMO

シングルクオートとダブルクオート、どちらを使うかですが、機能に違いはありませんので、好みでどちらを使っても構いません。日本語の「」と『』の違いのようなものです。

ただし、文字列リテラルの中でシングルクオートやダブルクオートを使いたいときに、外側のクオートと内側のクオートは、異なるものを使う必要があります。

```
var nickName = '田中将大選手のニックネームは"マー君"です。';
```

COLUMN

特殊な文字を表すエスケープシーケンス

文字列の中に、改行やタブなど、ディスプレイには見えない文字を含めたいときは、**エスケープシーケンス**という書き方を使います。ダブルクオートで囲んだ文字列リテラルの中で、文字列としてダブルクオートを使いたいときもエスケープシーケンスを使います。

▼おもなエスケープシーケンス

意味	エスケープシーケンス
タブ	¥t
改行	¥n
シングルクオート	¥'
ダブルクオート	¥"
¥(円記号)	¥¥

データの種類③

——論理値

SECTION

4

データの種類の3つ目は、論理値です。聞き慣れないかもしれませんが、要するに、二者択一のペアを表すデータです。

● 論理値とは？

論理値とは、正しい(真)か正しくない(偽)かを表す値で、取り得る値はこの2つのどちらかだけです。正しい(真)を表す値は **true**、正しくない(偽)を表す値は **false** です。

● 論理値の代入のしかた

変数に論理値データを代入する場合は、イコールに続けて、trueまたはfalseと書きます。

▼ 構文：論理値の代入文

変数名 = true (または false) ;

たとえば、「check」という変数を宣言して、trueという論理値データを代入するなら、このように書きます。

```
var check = true;
```

MEMO

論理値は別名、真偽値、真理値、ブール値、ブーリアン型などとも呼ばれます。文字列ではないので、クォートで囲んではいけません。文字列の「true」を代入したいときは、クォートで囲んで"true"と書きます。クォートがあるのとないのとでは、代入されるデータの種類が異なりますので、注意してください。





Lesson 4



算術演算子

数値の計算に使う「+」や「-」などの記号のことを、プログラミング用語で算術演算子といいます。

● 5 種類の算術演算子

数値の計算に使う算術演算子には、5つの種類があります。

演算子	読み方	機能
+	プラス	足す（加算）
-	マイナス	引く（減算）
*	アスタリスク	掛ける（乗算）
/	スラッシュ	割る（除算）
%	パーセント	余りを求める（除算の剰余）

ソースコードでの使い方はこのようになります。

```
a = 7 + 3; //10 が代入される
b = 7 - 3; //4 が代入される
c = 7 * 3; //21 が代入される
d = 7 / 3; //2 が代入される
e = 7 % 3; //1 が代入される
```

MEMO

1つの文の中に複数の算術演算子がある場合は、基本は先頭から順番に計算されますが、「*」「/」「%」があれば、それらが先に計算されます。



連結演算子

「+」記号を文字列に対して使うと、文字列と文字列をつなぎ合わせる連結演算子になります。

● 文字列の連結

複数の文字列を合体させることを「文字列の連結」といいます。文字列を連結するには、**連結演算子 (+)** を使います。

```
var fullName = '田中' + '将大';
```

これで変数 fullName には、「田中将大」という文字列が代入されます。
連結演算子 (+) を使えば、文字列が入った変数同士を連結することもできます。

```
var familyName = '田中';  
var firstName = '将大';  
var fullName = familyName + firstName;
```

この場合も、変数 fullName には、「田中将大」という文字列が代入されます。

数字の文字列同士を連結すると、計算はされずに、単純につなぎ合わされます。

```
var telephoneNumber = '03' + '5217' + '2400';
```

この場合、変数 telephoneNumber には「0352172400」という文字列が代入されます。



Lesson 4



コメントの書き方

コメントは、ソースコード中に自由に書き込めるメモのようなものです。ソースコードを読みやすくするためなどに使います。

● 2 種類のコメント

コメントは、ソースコード中に自由に書き込めるので、メモのように使えます。コメントの内容は、おもに処理の内容の簡単な説明です。あとからソースコードを読んだときに、どこにどんな処理が書いてあるのかがひと目で分かるようにしておきます。

コメントには、「行コメント」と「ブロックコメント」という2種類の書き方があります。どちらも、プログラムの実行時には無視されます。

● 行コメント

行コメントは、1 行のコメントです。「//」から「改行」までがコメントと見なされます。「//」は行頭に書いてもいいですし、行の途中に書いても構いません。

// 変数の宣言

```
var a = 2; // 変数の宣言と初期化
```

● ブロックコメント

ブロックコメントは、複数行にわたって使えるコメントです。「/*」から「*/」までがコメントと見なされます。「/*」は行頭に書いても、行の途中に書いても構いません。

/* ここからがコメント

変数の宣言 */

```
var a = 2; /* 変数の宣言と初期化 */
```



データ型

JavaScript では、データの種類のことをデータ型といいます。データ型には「単純なもの」と「複合したもの」と「特殊なもの」があります。

● 単純データ型

単純データ型というのは、1つの変数に1個のデータを入れる場合のデータの種類の、すでに学んだ数値、文字列、論理値の3つがあります。単純データ型は他にも、**プリミティブデータ型**、**プライマリデータ型**、**基本データ型**などとも呼ばれます。

● 複合データ型

JavaScript では、1つの変数に複数のデータを入れることもできます。その場合には**複合データ型**を使います。複合データ型というのは、データ1個1個の種類を指すのではなく、複数のデータをまとめておくための構造のことを指します。複合データ型には、このあと学ぶ配列とオブジェクトがあります。複合データ型は**参照型**とも呼ばれます。

● 特殊なデータ型

その他に、上の2つに含まれない特殊なデータ型があります。**null**と**undefined**です。このあとの1-6-3で学びます。



Day 1



Lesson 5

データの種類④——配列

このレッスンでは、1つの変数で複数のデータをまとめて管理できるしくみの1つである、配列について学びます。

- 1 変数に複数のデータを代入する方法
- 2 配列のしくみ
- 3 先頭の要素を削除する `shift()` メソッド
- 4 先頭に要素を追加する
`unshift()` メソッド
- 5 末尾の要素を削除する `pop()` メソッド
- 6 末尾に要素を追加する
`push()` メソッド
- 7 要素を削除する `splice()` メソッド
- 8 配列の長さを調べる `length` プロパティ

変数に複数のデータを代入する方法

SECTION

1

1つの変数には、複数のデータを代入することもできます。複数のデータを代入するには、配列というしくみか、オブジェクトというしくみを使います。

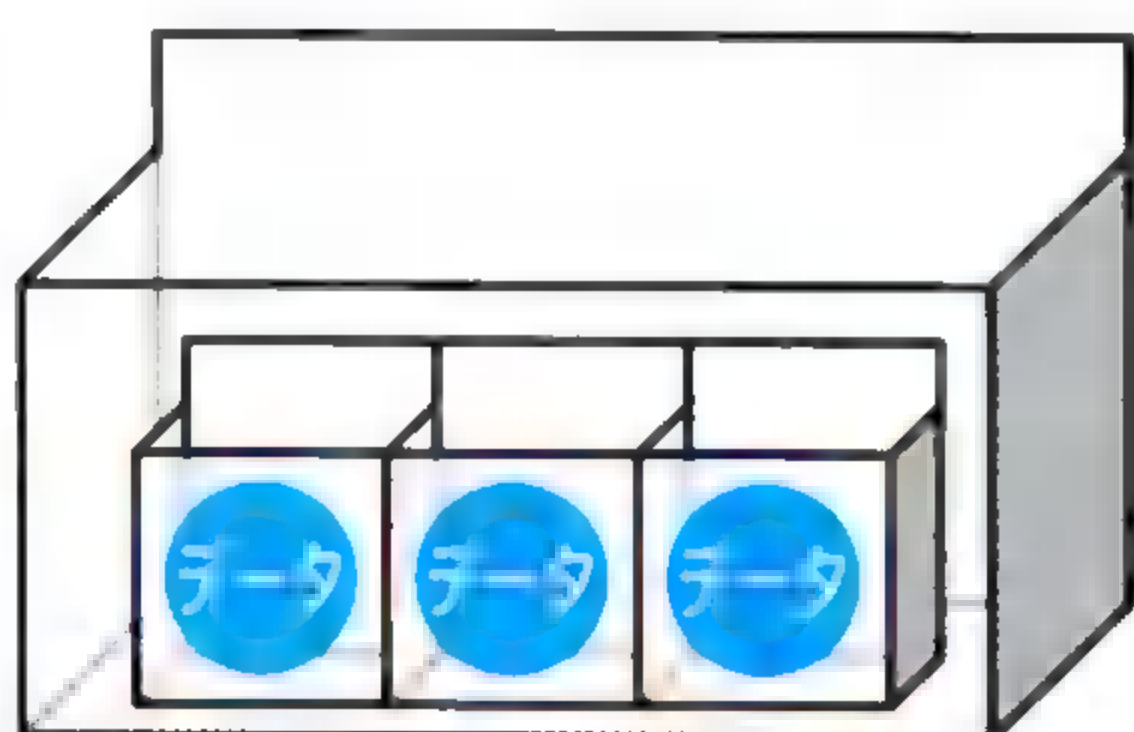
● 1つの変数で複数のデータを管理できる

前のレッスンでは、1つの変数で1つのデータを管理する方法を学びました。しかし、データの数が増えてくると、それぞれのデータに変数を1つずつ用意していたのでは、管理がわずらわしくなってきます。関連する複数のデータは、まとめて1つの変数で管理できれば効率的です。

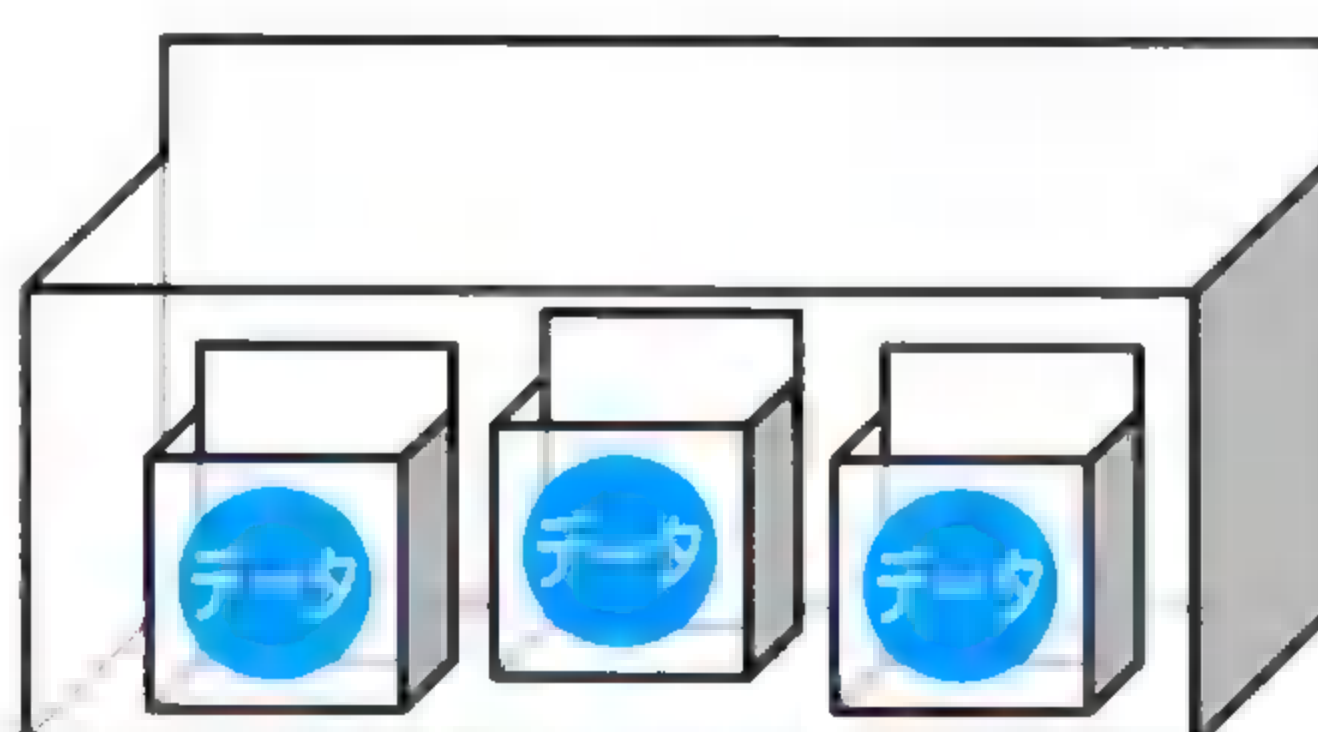
JavaScriptでは、1つの変数で複数のデータをまとめて管理するしくみが用意されています。1つは**配列**、もう1つは**オブジェクト**です。

● 配列とオブジェクトのイメージ

詳しい説明はあとに譲るとして、まずは配列とオブジェクトを簡単なイメージで表してみます。それぞれどのようなものか、だいたいのイメージをつかんでおいてください。



配列のイメージ



オブジェクトのイメージ



配列のしくみ

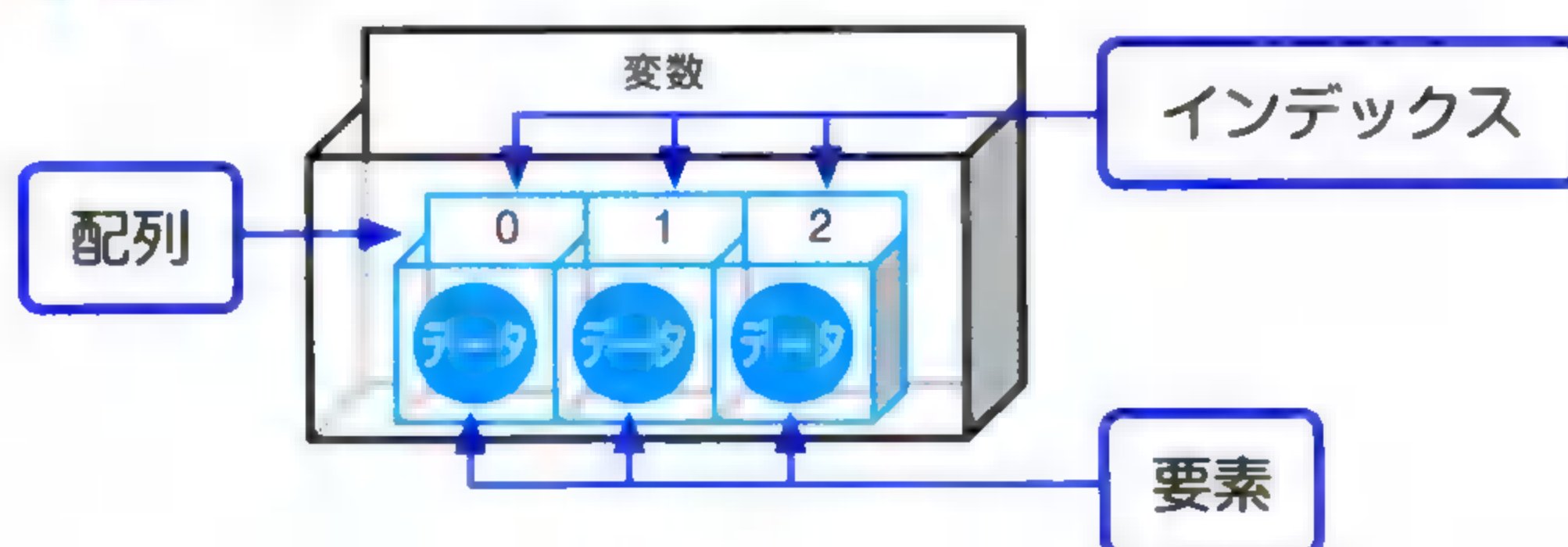
配列を使うと、1つの変数に複数のデータを代入することができます。配列は、同じ種類のデータをまとめて扱うのに向いています。



● 配列とは？

配列とは、1つの変数に複数のデータをまとめて入れられるしくみの1つです。変数を1つの箱だとすると、その箱の中に、別の箱がいくつもつながって入っているようなイメージです。この箱の中の箱全体を**配列**といいます。

▼ 配列のしくみ



配列のそれぞれの箱を**要素**といい、この中にデータを代入できます。

要素には自動的に、先頭から順番に番号が振られていて、この番号を**インデックス**（または添え字）といいます。

インデックスは「0」から始まります。1番目の要素のインデックスは「0」、2番目の要素のインデックスは「1」、3番目の要素のインデックスは「2」となります。

● 配列の作成方法

変数に複数のデータを配列として代入するには、**大カッコ([])**と**カンマ(,)**を使って、このように書きます。

▼ 構文：配列の代入文

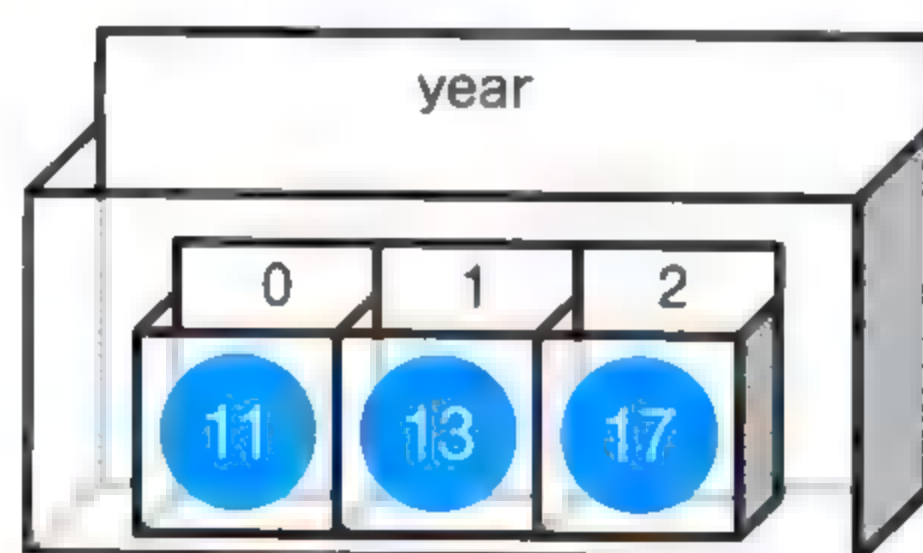
変数名 = [データ 1, データ 2, データ 3] ;

これで変数の中には配列が用意され、データは先頭から順番に要素に代入されます。配列の各要素には、数値、文字列、論理値などを代入できます。

● 配列に数値を代入してみる

たとえば、変数の宣言と同時に3つの数値を配列として代入する場合は、このように書きます。

```
var year = [11, 13, 17];
```

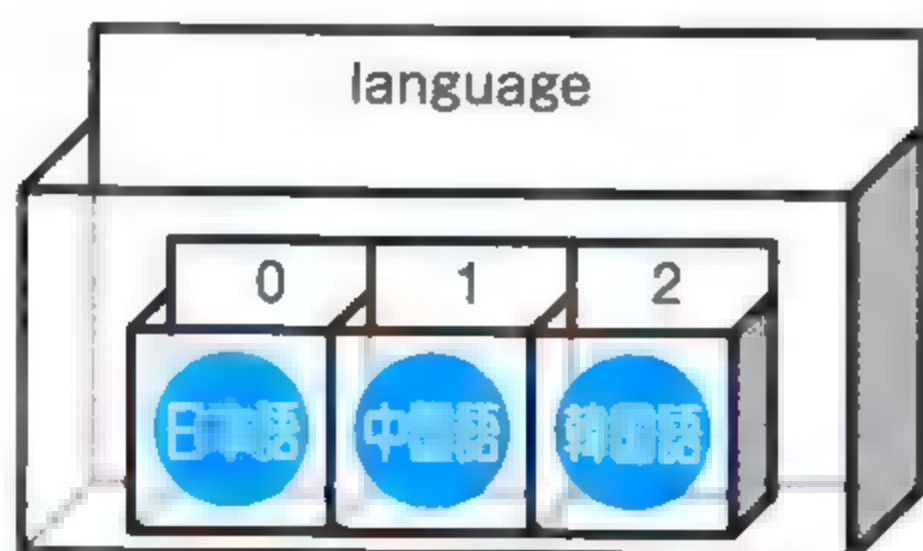


配列に数値を代入

● 配列に文字列を代入してみる

3つの文字列を配列として代入する場合は、それぞれの文字列をクオートで囲みます。たとえば、すでにある language という変数に「日本語」「中国語」「韓国語」という文字列を要素として持つ配列を代入するには、このように書きます。

```
language = [ '日本語', '中国語', '韓国語' ] ;
```

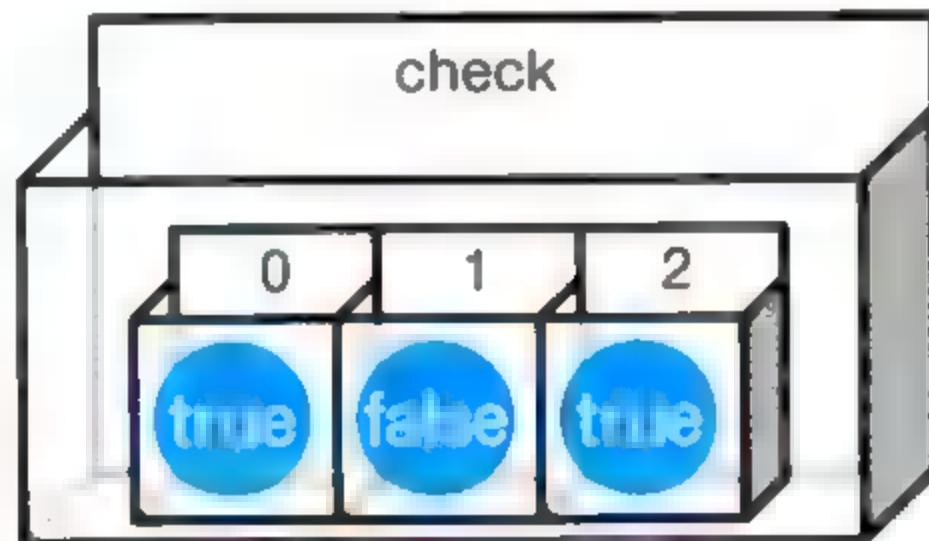


配列に文字列を代入

● 配列に論理値を代入してみる

3つの論理値を配列として代入するには、trueやfalseをカンマで区切って書きます。

```
check = [true, false, true];
```



配列に論理値を代入

MEMO

同じ配列に、異なる種類のデータを代入することもできます。たとえば、インデックスが0の要素に数値、1の要素に文字列、2の要素に論理値を代入してもかまいません。ただ、配列は同じ種類のデータをまとめて管理するのに適したデータ型ですので、実際のプログラムでは、異なるデータ型のデータを格納することはあまりありません。

● 配列の要素を指定してデータを代入するには？

配列に入れてあるデータは、1つずつ個別に出し入れできます。配列の要素を指定すれば、データを代入したり、参照したりできます。要素を指定して代入や参照をするときには、配列が入っている**変数名とインデックス**を使います。

▼ 配列の各要素の指定のしかた

配列が入った変数名 [インデックス]

たとえば、変数yearの中にある配列の先頭から2番目の要素に、「20」という数値を代入したいときには、このように書きます。

```
year[1] = 20;
```

● 配列の要素を参照するには？

配列の要素を参照するには、先ほどと同じく、変数名と大カッコとインデックスを使います。たとえば、aという変数に要素のデータを代入したいときにはこのように書きます。

```
a = year[1];
```

MEMO

1つの変数の中には、配列は1つしか入れられません。よって、配列が入っていれば、変数=配列ということになります。今後、「配列が入っている変数」は略して「配列」とだけ呼ぶことにします。「配列名」といった場合は、「配列が入っている変数名」を意味します。

MEMO

配列の要素には、数値、文字列、論理値だけでなく、配列やオブジェクト、さらには関数でさえも代入することもできます（オブジェクトについては1-6、関数については2-3で説明します）。

MEMO

変数に配列を代入するには、new Arrayという構文を使うこともできます（2-4-5参照）。

Lesson 5



先頭の要素を削除する shift() メソッド

配列はただデータを代入したり参照したりするだけではなく、あとから要素を削除したり、追加したり、いろいろな使い方ができます。

● 先頭の要素を削除するには？

配列の先頭の要素を取り除くには、**shift() メソッド**を使って、このように書きます（メソッドとは何かについては1-6-1で説明します）。

▼ 構文：shift() メソッド

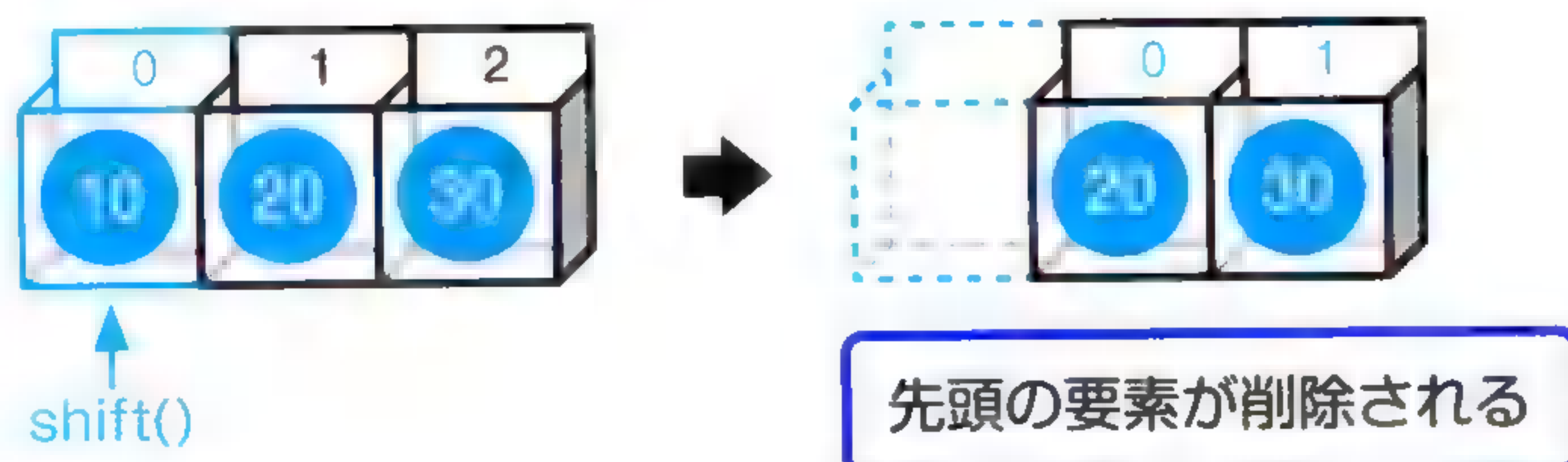
配列名 . **shift()**

● shift() メソッドを使ったサンプル

以下のソースコードは、shift() メソッドを使ったサンプルです。配列hにデータを代入し（1行目）、先頭の要素を削除したあと（2行目）、各要素のデータを警告ダイアログボックスで表示しています（3行目）。

```
1 var h = [10, 20, 30];  
2 h.shift();  
3 window.alert(h[0] + ',' + h[1] + ',' + h[2]);
```

▼ shift() メソッドの動作イメージ





先頭に要素を追加する unshift() メソッド

配列の先頭に要素を追加するには、unshift() メソッドというものを 사용합니다。

● 先頭に要素を追加するには？

配列の先頭に要素を追加してデータを代入するには、**unshift() メソッド**を使って、このように書きます。

▼ 構文：unshift() メソッド

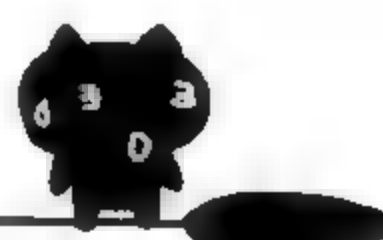
配列名.unshift(代入するデータ)

● unshift() メソッドを使ったサンプル

以下のソースコードは、unshift() メソッドを使ったサンプルです。配列hにデータを代入し(1行目)、先頭の要素を追加して「10」を代入したあと(2行目)、各要素のデータを警告ダイアログボックスで表示しています(3行目)。

```
1 var h = [20, 30];  
2 h.unshift(10);  
3 window.alert(h[0] + ',' + h[1] + ',' + h[2]);
```

▼ unshift() メソッドの動作イメージ



Lesson 5



末尾の要素を削除する pop() メソッド

配列の末尾の要素を削除するには、pop() メソッドというものを
使います。

● 末尾の要素を除去するには

配列の末尾の要素を取り除くには、**pop() メソッド**を使って、このように書きます。

▼ 構文：pop() メソッド

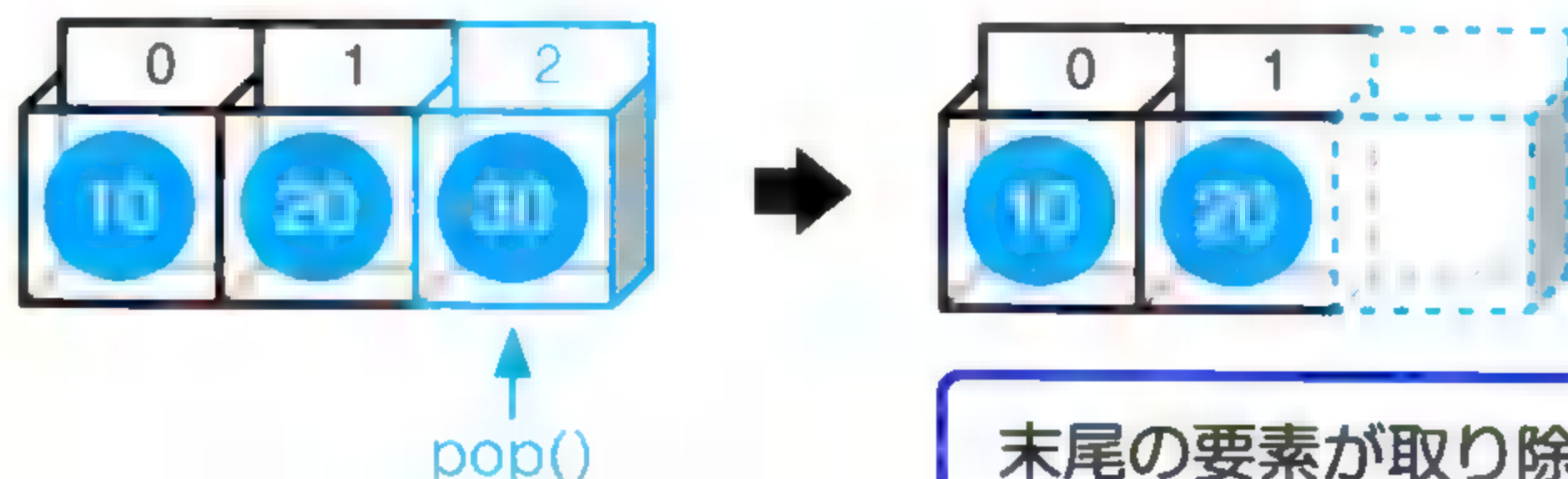
配列名 .pop()

● pop() メソッドを使ったサンプル

以下のソースコードは、pop() メソッドを使ったサンプルです。配列 h にデータを代入し (1 行目)、末尾の要素を取り除いたあと (2 行目)、各要素のデータを警告ダイアログボックスで表示しています (3 行目)。

```
1 var h = [10, 20, 30];
2 h.pop();
3 window.alert(h[0] + ',' + h[1] + ',' + h[2]);
```

▼ pop() メソッドの動作イメージ



末尾の要素が取り除かれる



末尾に要素を追加する push() メソッド

配列の末尾に要素を追加するには、push() メソッドというものを
使います。

● 末尾に要素を追加するには？

配列の末尾に要素を追加してデータを代入するには、**push() メソッド**を使って、
このように書きます。

▼ 構文：push() メソッド

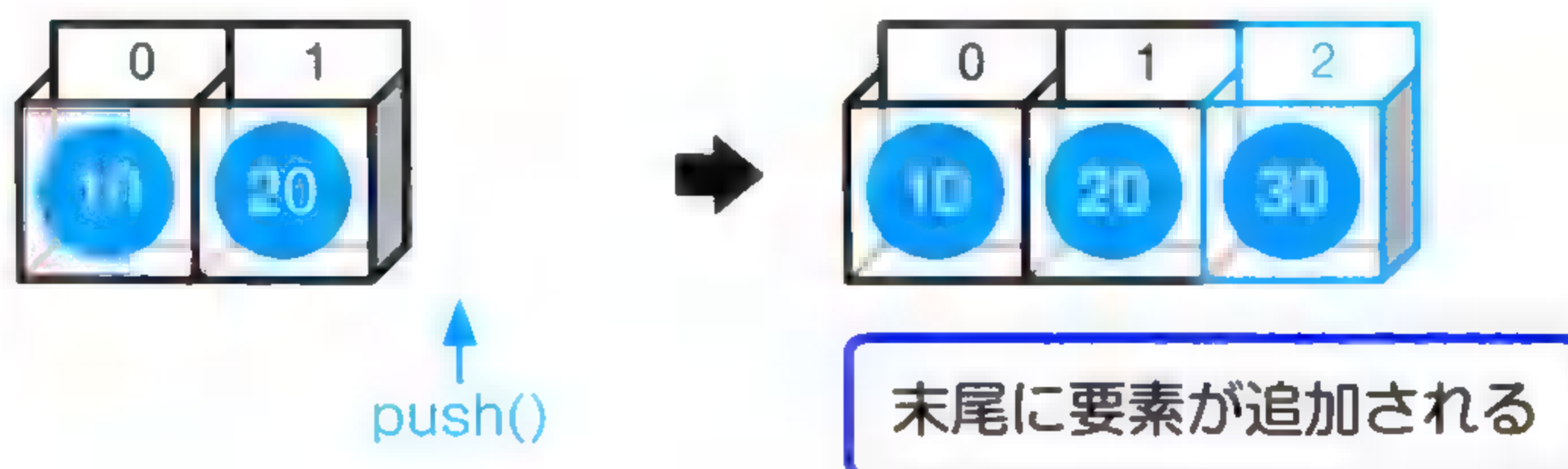
配列名 .push(代入するデータ)

● push() メソッドを使ったサンプル

以下のソースコードは、push() メソッドを使ったサンプルです。配列hにデータを
代入し (1 行目)、末尾に要素を追加して「30」を代入したあと (2 行目)、各要素のデー
タを警告ダイアログボックスで表示しています (3 行目)。

```
1  var h = [10, 20];  
2  h.push(30);  
3  window.alert(h[0] + ',' + h[1] + ',' + h[2]);
```

▼ push() メソッドの動作イメージ



Lesson 5



要素を削除する splice() メソッド

配列の要素を削除するには、splice() メソッドというものを使います。

● 配列の要素を削除するには？

配列のどれか要素を指定して削除するには、**splice() メソッド**を使って、このように書きます。

▼ 構文：splice() メソッド

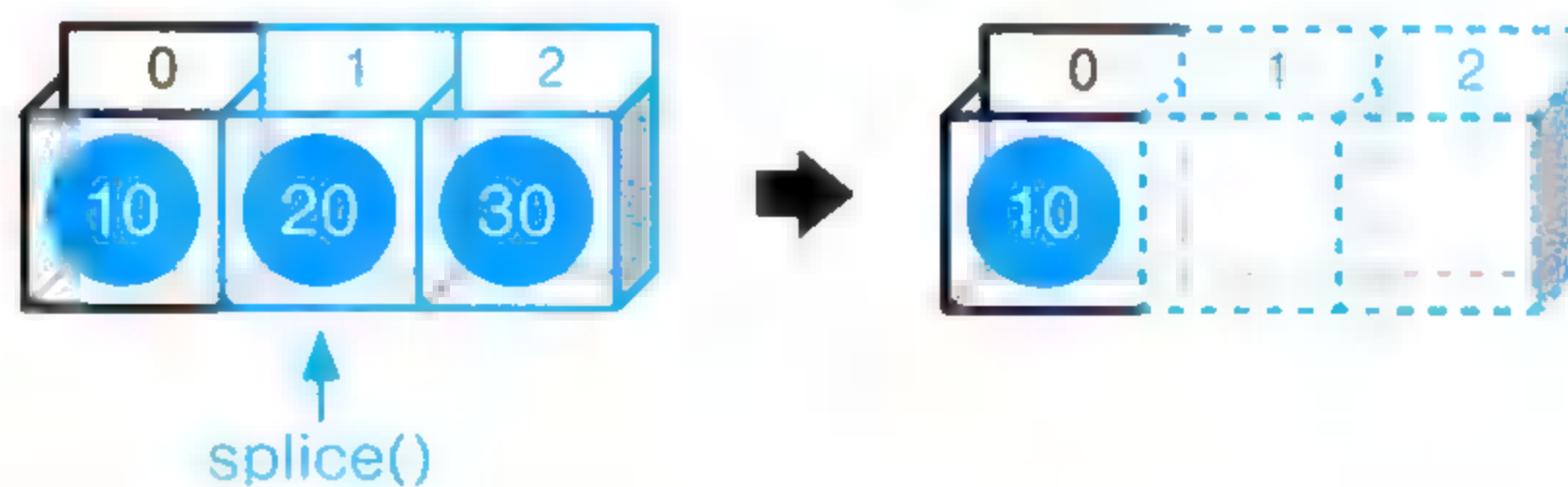
配列名.**splice**(インデックス, 削除する要素の数)

● splice() メソッドを使ったサンプル

以下のソースコードは、splice() メソッドを使ったサンプルです。配列hにデータを代入し（1行目）、2番目の要素から2つの要素を削除したあと（2行目）、各要素のデータを警告ダイアログボックスで表示しています（3行目）。

```
1 var h = [10, 20, 30];  
2 h.splice(1, 2);  
3 window.alert(h[0] + ',' + h[1] + ',' + h[2]);
```

▼ 構文：splice() メソッドの動作イメージ



インデックス1の要素を含め、
2つ要素が削除された



配列の長さを調べる length プロパティ

配列の長さを調べるには、length プロパティというものを使います。

● 配列の長さ（要素数）を調べるには？

要素を追加したり削除したりしているうちに、いまいったい要素の数はいくつあるのか、わからなくなることがあります。そんなときは配列の長さを調べます。配列の長さ、すなわちその配列にはいくつ要素があるのかを調べるには、**length プロパティ**を使います（プロパティについては1-6で説明します）。

▼ 構文：length プロパティ

配列名.**length**

● length プロパティを使ったサンプル

以下のソースコードは、length プロパティを使ったサンプルです。配列hにデータを代入し（1行目）、要素の数を警告ダイアログボックスで表示しています（3行目）。

```
1 var h = [10, 20, 30];  
2 window.alert(h.length);
```

▼ length プロパティの動作イメージ



要素の数を調べられる



Day 1



Lesson 6

データの種類⑤ ——オブジェクト

このレッスンでは、1つの変数で複数のデータを管理できるもう1つのしくみであるオブジェクトについて学びます。

- 1 オブジェクトのしくみ
- 2 オブジェクトの使い方
- 3 2つの特殊なデータ型

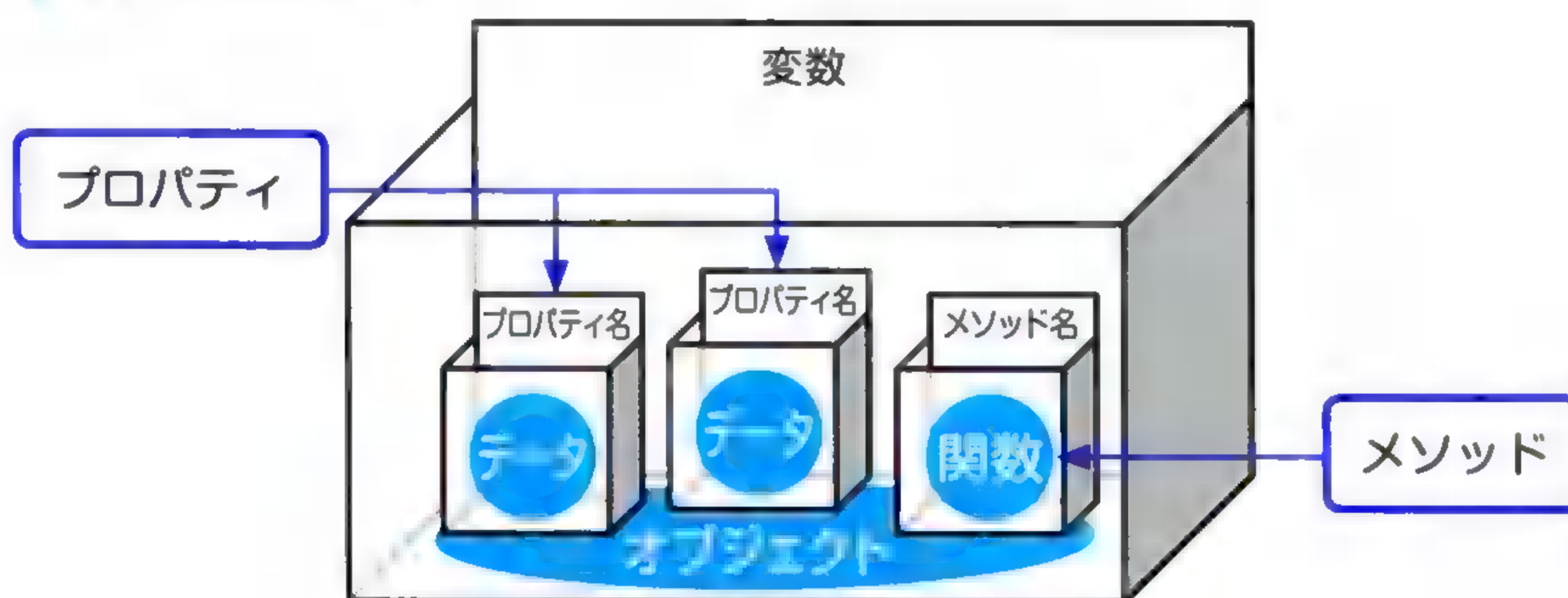
オブジェクトのしくみ

オブジェクトとは、変数の中に入った変数のことです。1つの変数で、違う種類の複数のデータをまとめて管理するのに適しています。

● オブジェクトとは？

変数には、データを直接入れる以外に、変数も入れることができます。オブジェクトというのは、ひとことで言えば、変数の中に入れられた変数、すなわち「変数 in 変数」のことです。「変数 in 変数」はいくつでも作ることができ、全部まとめて1組のオブジェクトです。

▼ オブジェクトのしくみ



変数の中の変数は、「変数」とは呼びません。**プロパティ**といいます。プロパティにはそれぞれ**プロパティ名**を付けて、データを入れて管理します。プロパティ名の付け方は、変数名と同じ規則に従います。

また、プロパティには、データではなく、関数を入れることもできます。関数が入ったプロパティのことを、**メソッド**といいます（関数は2-3で説明します）。



● 配列とオブジェクトの違い

1つの変数で複数のデータを管理するしくみといえば、前のレッスンで配列を学びました。配列と違うのは、配列の要素箱は1列に並んでいて、自動で先頭から順番に番号が振られていたのに対して、オブジェクトのプロパティ箱は順番には並んでいませんし、番号も振られていません。

▼ 配列とプロパティの違い



● オブジェクトの作成方法

変数の中にオブジェクトを作るには、このような書き方をします。

▼ 構文：オブジェクトの代入文

```
変数名 = {
  プロパティ名1: データ1,
  プロパティ名2: データ2,
  プロパティ名3: データ3
};
```

プロパティ名はプログラマが考えて付けます。名前の付け方の規則は、変数名と同じです。こうすることで、変数の中にプロパティが作られ、その中にデータが代入されます。

MEMO

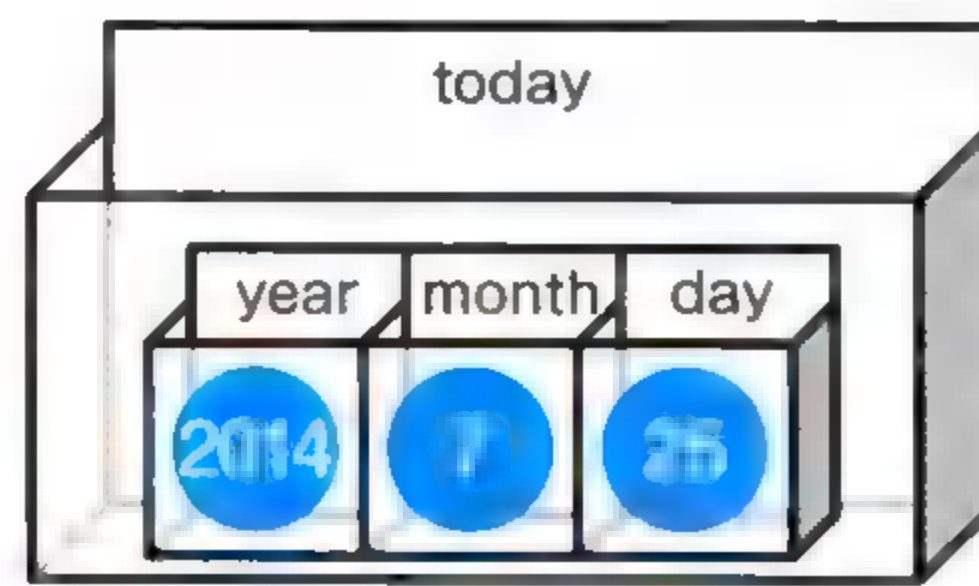
{や:,のあとに半角スペースや改行を入れていますが、見た目の問題ですので、改行やスペースなしで書いてもかまいません。

たとえば、変数todayの中に、プロパティを3つ持つオブジェクトを作って、それぞれ「年」「月」「日」を表す数値を代入したいとします。この場合は、ソースコードはこのような書き方です。

```

1  var today = {
2      year: 2014,
3      month: 7,
4      day: 25
5  };

```



オブジェクトが作られた

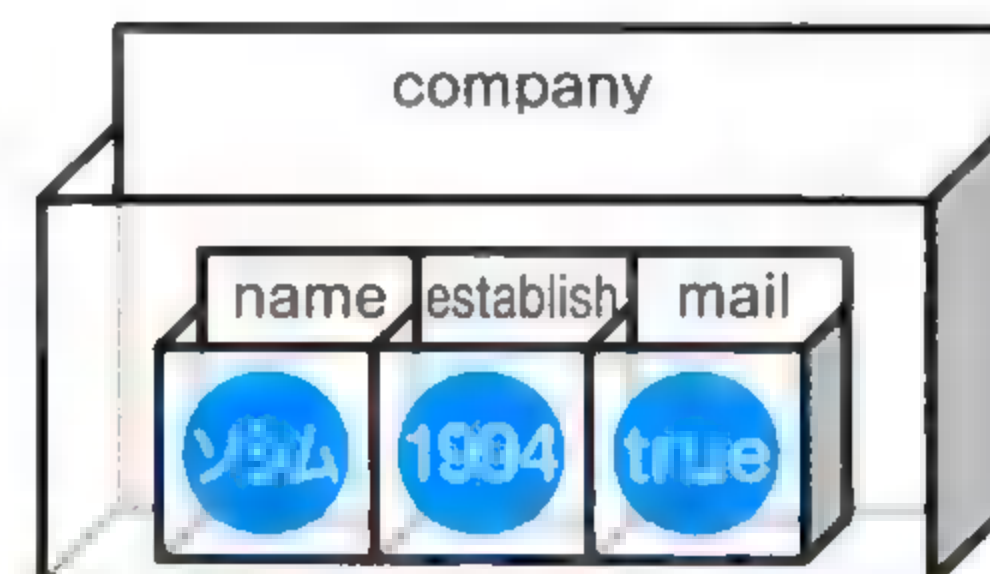
● プロパティごとに異なるデータも代入できる

オブジェクトは、プロパティごとに異なる種類のデータも代入できます。

```

1  var company = {
2      name: 'ソシム', // 文字列を代入
3      establish: 1994, // 数値を代入
4      mail: true // 論理値を代入
5  };

```



それぞれのプロパティに異なる種類の値を代入

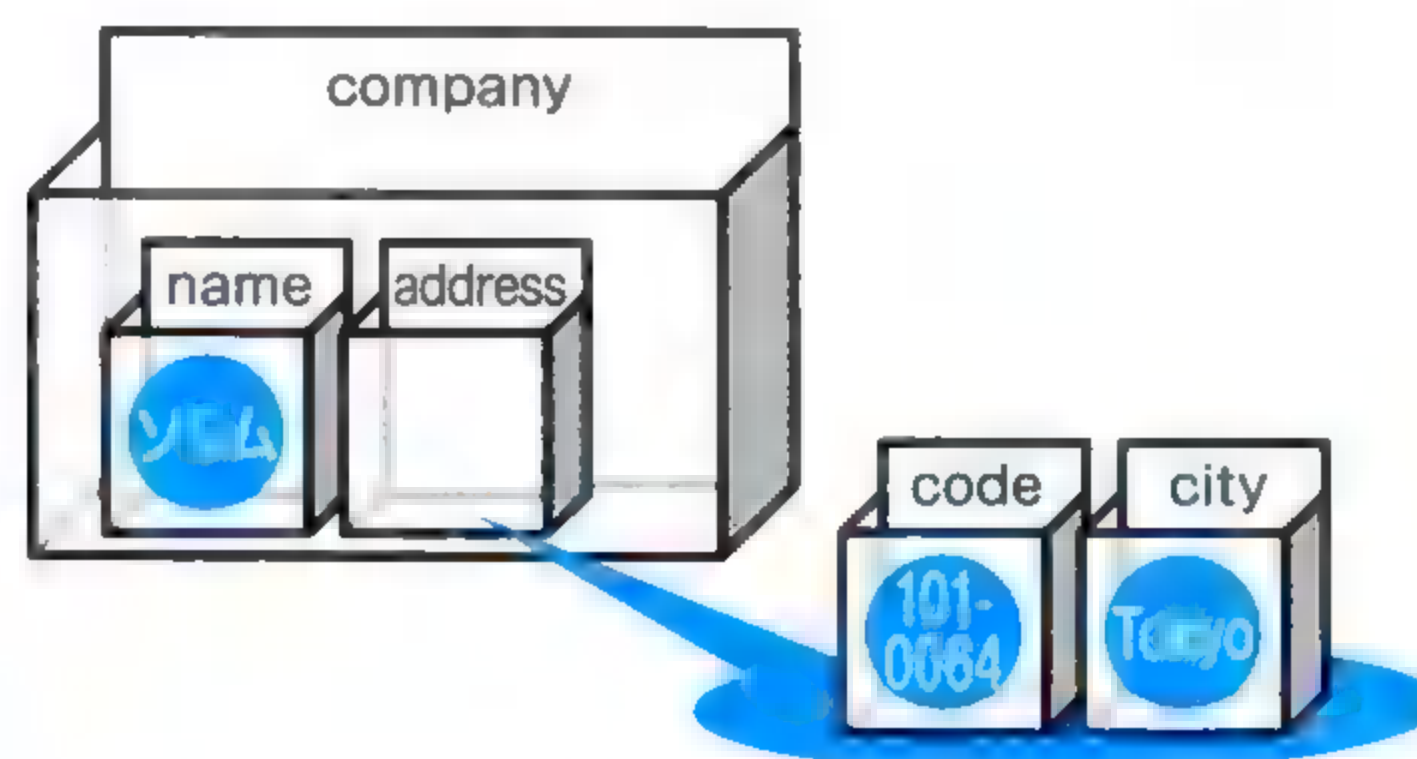
● オブジェクトは入れ子にできる

オブジェクトのプロパティの中には、直接データを入れる以外に、さらに別のオブジェクトを入れることもできます。たとえば、address プロパティの中に、code プロパティと city プロパティで構成されるオブジェクトを作るには、このように書きます。

```

1  var company = {
2      name: 'ソシム',
3      address: {
4          code: '101-0064',
5          city: 'Tokyo'
6      }
7  };

```

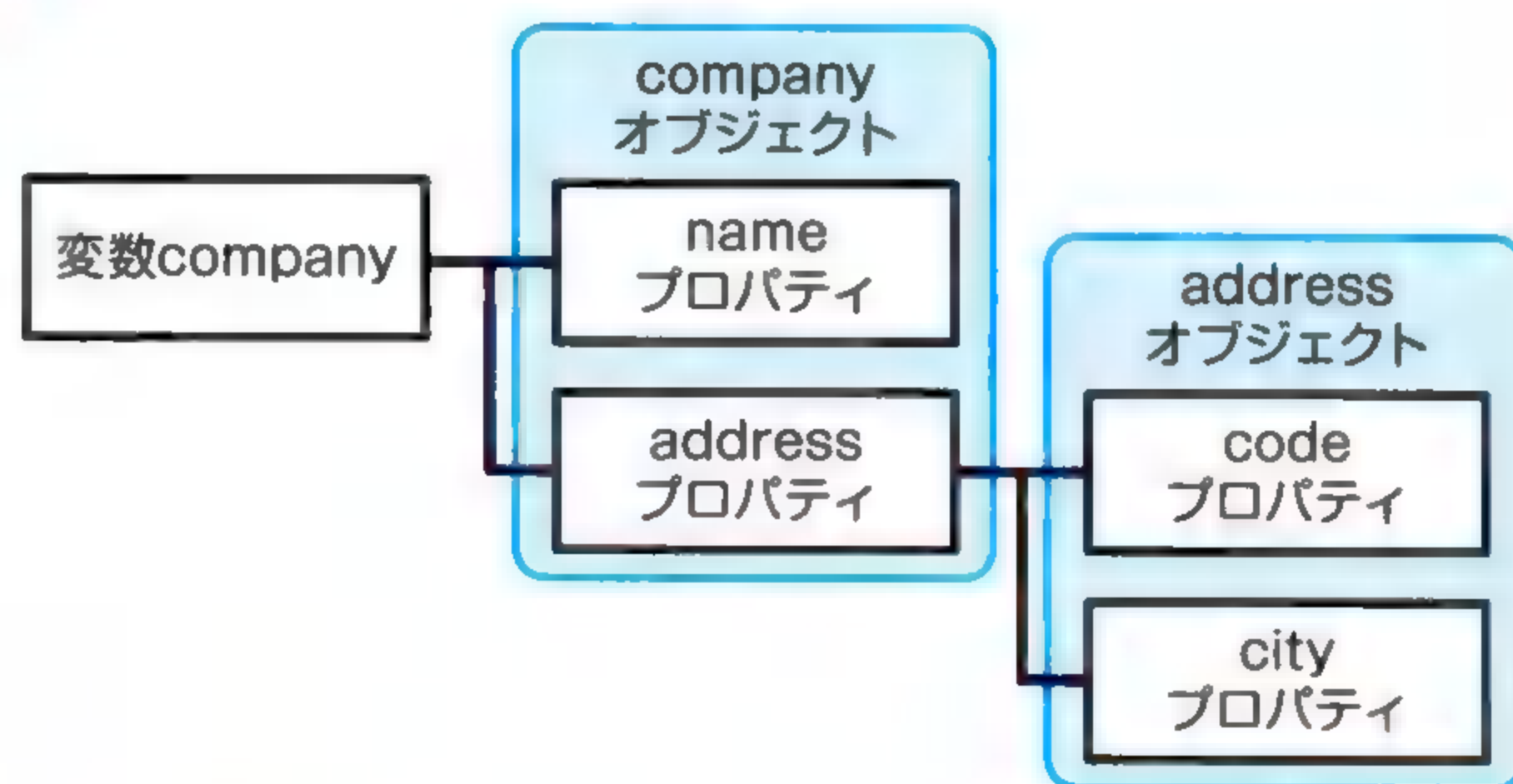


入れ子になったオブジェクトのイメージ

● オブジェクトを図解するには樹形図を使う

入れ子になったオブジェクトの構造は、よく樹形図（ツリー図）を使って表されます。たとえば、変数 `company` に入っているオブジェクトは、樹形図で表すとこのようになります。

▼ オブジェクトを表す樹形図



MEMO

オブジェクトの階層構造や、それを表した樹形図のことを、オブジェクトツリーということがあります。

オブジェクトの名前は、そのオブジェクトが入っている変数の名前や、プロパティの名前を使います。

たとえば、`company` という変数に入っているオブジェクトの名前（オブジェクト名）は、`company` になります。また、`address` というプロパティに入っているオブジェクトの名前は `address` になります。

MEMO

1つの変数（プロパティ）の中には、オブジェクトは1組しか入れられません。よって、オブジェクトが入っていれば、変数（プロパティ）＝オブジェクトということになります。

今後、「オブジェクトが入っている変数（プロパティ）」は略して「オブジェクト」とだけ呼ぶことにします。「オブジェクト名」といった場合は、「オブジェクトが入っている変数名（あるいはプロパティ名）」を意味します。

オブジェクトの使い方

プロパティに入っている値を参照したり、上書きしたり、あとからプロパティを追加したりする方法を学びます。

● プロパティの値を参照するには？

プロパティの値を参照するには、オブジェクト名とピリオド(.)とプロパティ名を使ってこのように書きます。

▼ プロパティの値の参照方法

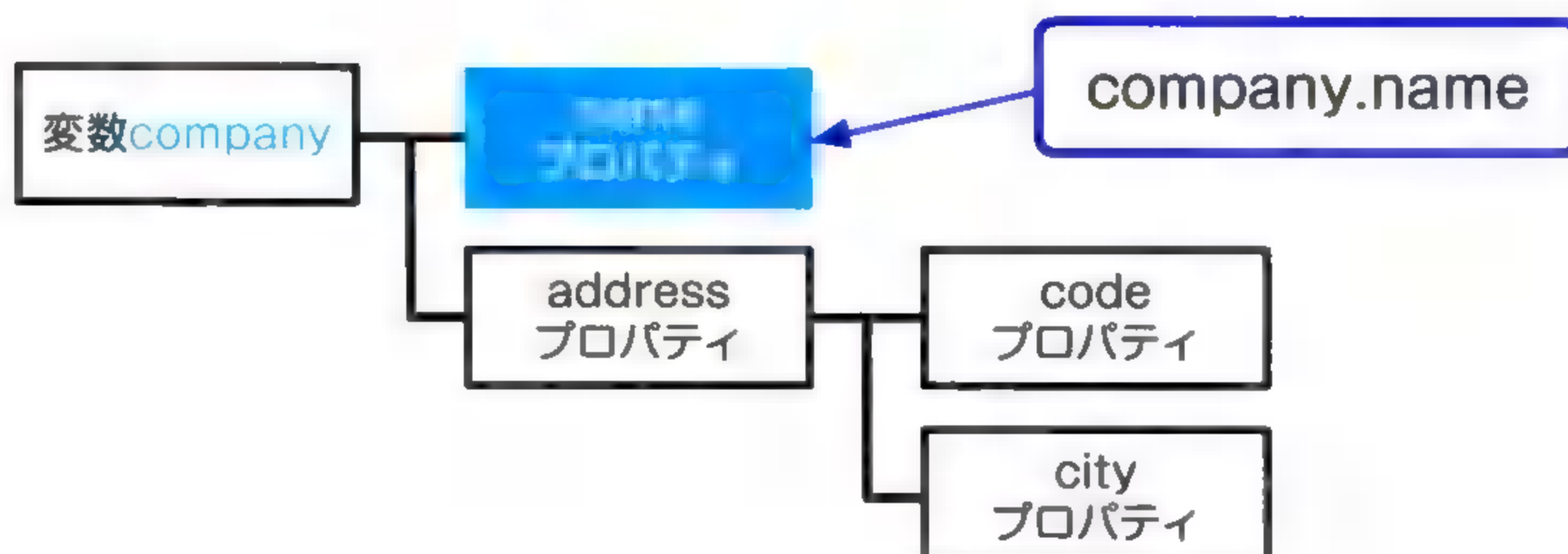
オブジェクト名 . プロパティ名

たとえば、company オブジェクトの name プロパティの値を参照して、警告ダイアログボックスに表示するソースコードは、このように書きます。

```

1  var company = {
2      name: 'ソシム',
3      address: {
4          code: '101-0064',
5          city: 'Tokyo'
6      }
7  };
8  window.alert(company.name);

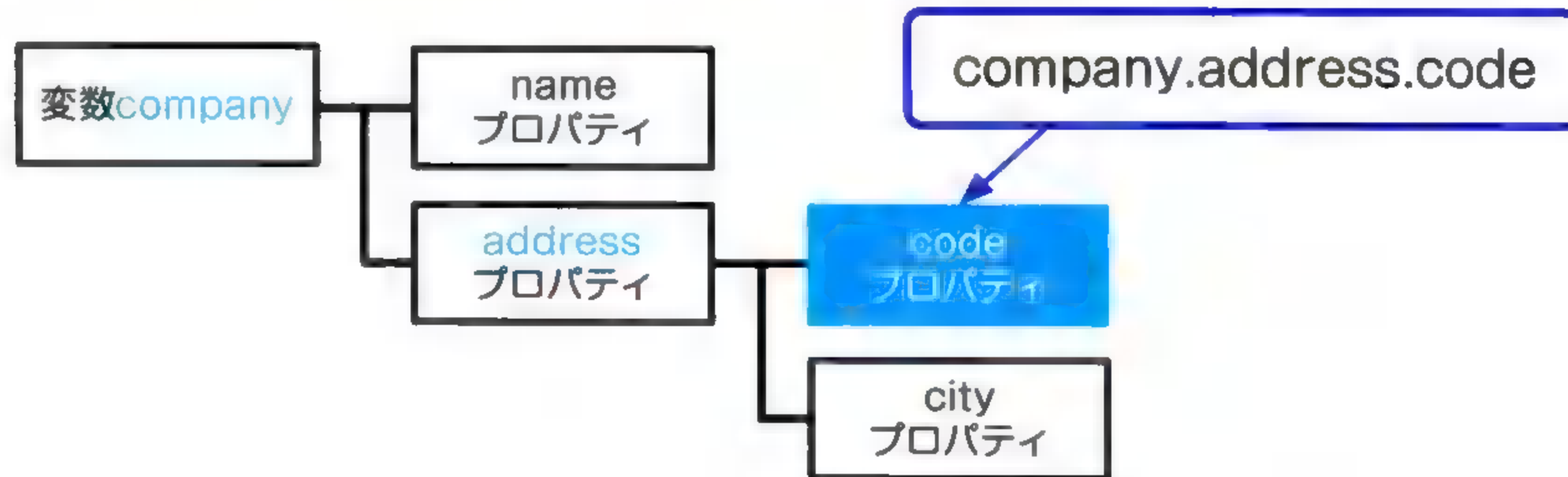
```



入れ子になっているオブジェクトのプロパティ値を参照するには、ピリオド (.) でプロパティ名をつなげていきます。

たとえば、company オブジェクトの address プロパティに入っている、address オブジェクトの code プロパティの値を参照して、警告ダイアログボックスに表示するには、このように書きます (8 行目に追加)。

```
8 window.alert(company.address.code);
```



● プロパティのデータを上書きするには？

プロパティに入っているデータを、別のデータで上書きするには、先ほどと同じく、ピリオド (.) を使ってプロパティを指定して、新たなデータを代入します。

```
8 company.name = '任天堂';
9 company.address.code = '601-8116'
10 company.address.city = 'Kyoto'
```

● あとからプロパティを追加するには？

「オブジェクト名. プロパティ名」の書き方を使って、存在していないプロパティ名を指定してデータを代入すれば、新しいプロパティを作成して追加することができます。

たとえば、company オブジェクトに yomigana という新たなプロパティを追加して、'ニンテンドウ' という文字列を代入するには、このように書きます。

```
11 company.yomigana = 'ニンテンドウ';
```

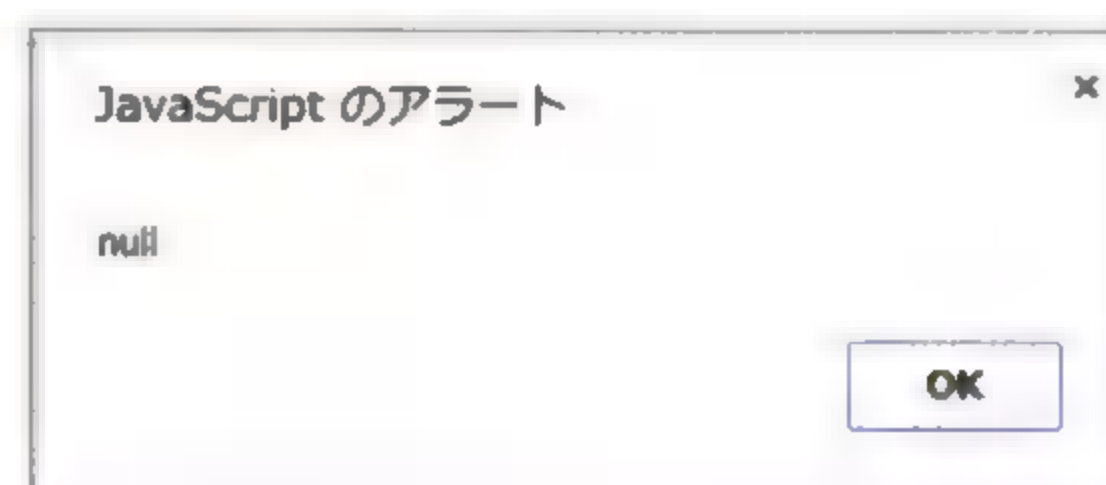
2つの特殊なデータ型

JavaScript のデータの種類には、これまで学んだ数値、文字列、論理値、配列、オブジェクト以外に、もう2つ、特殊なデータ型があります。最後に、その2つについて説明しておきます。

● 値がないことを表す null

null (ヌル) は、変数やプロパティの中に、**データがない**ことを表す値です。たとえば、以下のようなオブジェクトを作成する際に、電話がない会社の場合は、phone プロパティに null を代入しておく、というように使います。

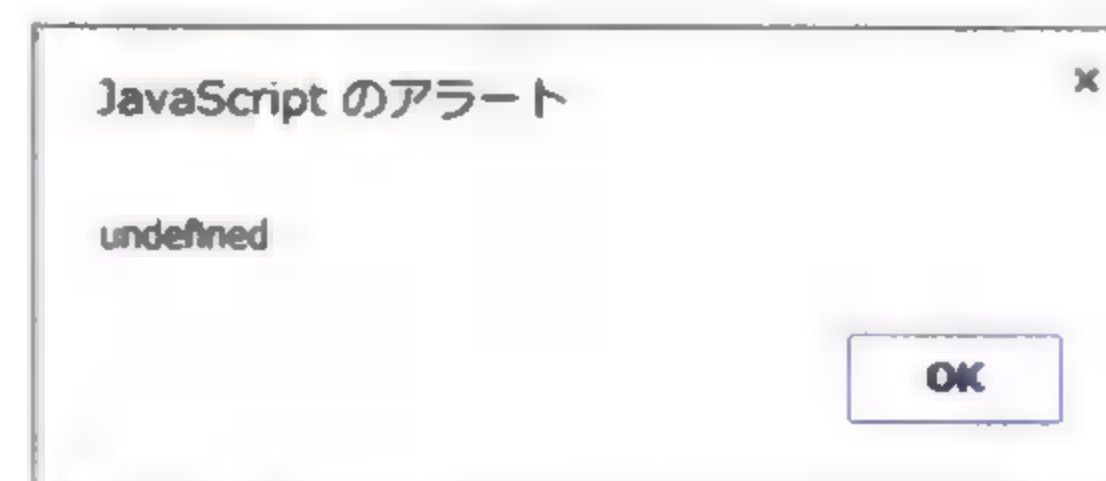
```
1 var company = {  
2   name: 'ソシム',  
3   phone: null  
4 };  
5 window.alert(company.phone);
```



● 存在しない変数、要素、プロパティを表す undefined

undefined (アンデファインド) は、存在していない変数や、要素、プロパティを使おうとしたときに表示される値です。たとえば、company オブジェクトには url というプロパティは存在しませんので、company.url を表示すると、undefined になります。

```
5 window.alert(company.url);
```



Day 2



Lesson 1

条件文

このレッスンでは、処理手順の順番を変えたいときに使う条件文のしくみと使い方を学びます。

- 1 条件文① ～ 目次
- 2 条件文② ～ if ～ else 文
- 3 条件文③ ～ if ～ else if ～ else 文
- 4 条件文④ ～ switch 文
- 5 条件の書き方

Lesson 1



条件文①——if 文

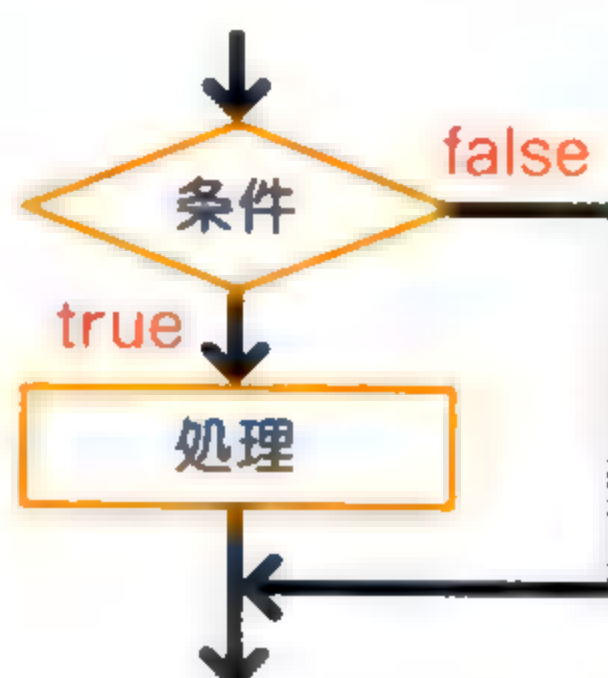
1 つ目の条件文は if 文です。if は英語で「もし～なら」という意味ですが、JavaScript でも同じ意味合いで使われています。

● if 文の書き方

if 文は、条件によって、ある処理を実行するかしないか選択して、処理の流れを分岐させたいときに使う構文です。

▼ 構文：if 文

```
if (条件) {  
    処理  
}
```



条件が true のときは
処理を実行する

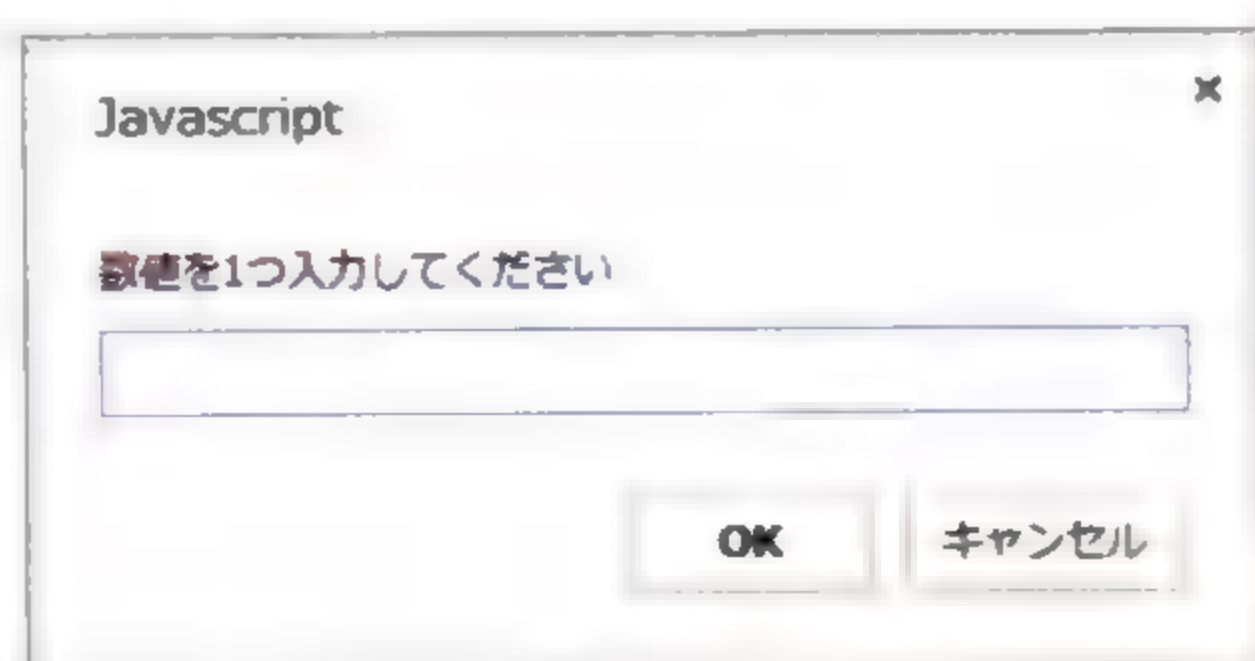
条件が成り立つとき (true のとき) は、処理を実行します。逆に、条件が成り立たないとき (false のとき) は、処理を実行せずにスキップします。

● if 文のサンプルプログラム

サンプルプログラムを使って、実際に if 文の動作を確認してみましょう。以下のソースコードを入力して、実行してみてください。

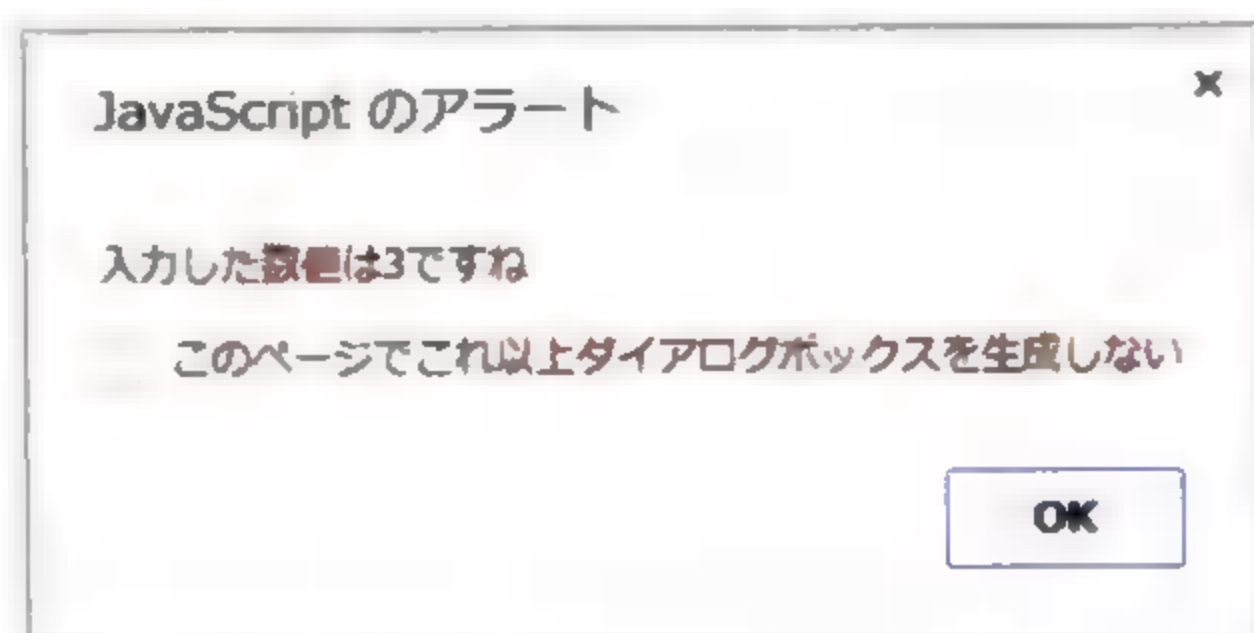
```
1  var n = window.prompt(' 数値を 1 つ入力してください ');  
2  if (n == 3) {  
3      window.alert(' 入力した数値は 3 ですね ');  
4  }
```

実行すると、入力ダイアログボックスが表示されます。



入力欄に「3」と入力

この入力欄に「3」と入力すると、警告ダイアログボックスが表示されます。「OK」ボタンをクリックすれば、プログラムは終了します。



ダイアログボックスが表示される

MEMO

この本の学習中は、チェックボックスはクリックしないでください。

● サンプルプログラムの解説

まず、入力ダイアログボックスの入力欄に入力された数値が、変数 n に代入されます (1行目)。if文の条件は「 $n == 3$ 」、すなわち「 n は3と等しい」です (2行目)。変数 n の値が3なら、「 n は3と等しい」ので条件が成り立ち (true)、警告ダイアログボックスが表示されます (3行目)。

もし変数 n が3でなければ、条件が成り立たない (false) ので、ダイアログボックスは表示されずに、プログラムは終了となります。3以外を入力して、試してみてください (条件の詳しい書き方は、2-1-5で説明します)。

COLUMN**ブロック**

条件のあとの中カッコ ($\{$ と $\}$) で囲まれた部分のことを、**ブロック** (またはコードブロック) といいます。ブロックの中には、複数の文を処理として書くことができます。

処理にあたる文が1つのときは、中カッコを省略しても大丈夫ですが、文が複数あるときは、必ず中カッコで囲んで、処理をブロックにしてください。

ブロックの中の文は、改行ごとにインデントすると見やすくなります。ブロックの終わり ($\}$ のあと) には、セミコロンは書きません。

条件文②

——if ～ else 文

SECTION

2

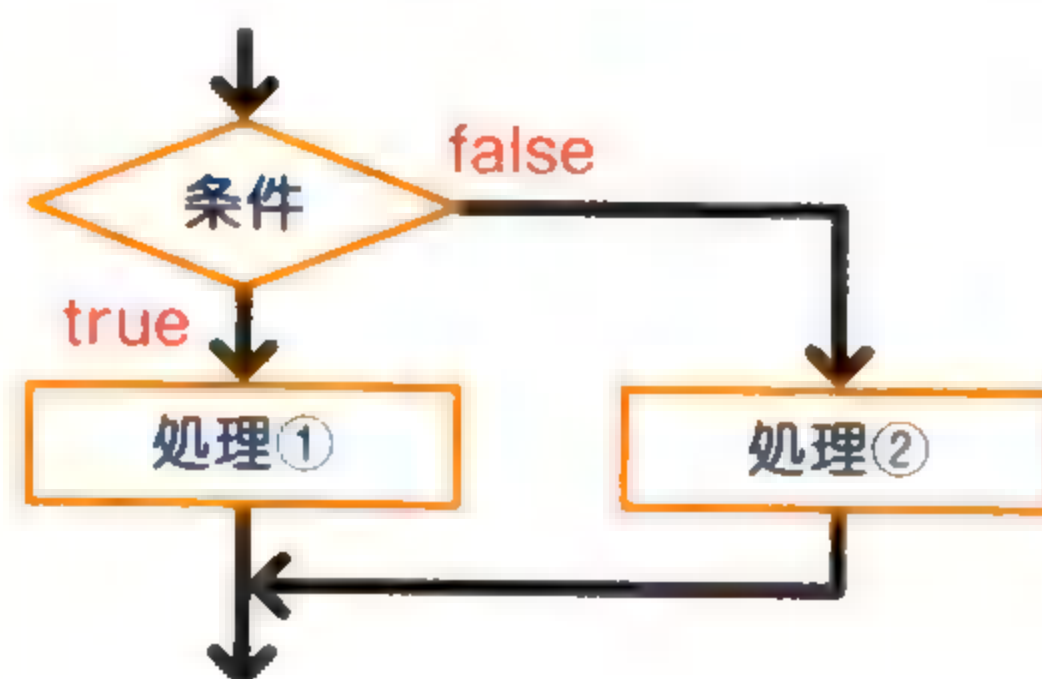
2つ目の条件文は、if ～ else (イフ エルス) 文です。else は英語で「それ以外の場合は」という意味です。

● if ～ else 文の書き方

if ～ else 文は、if 文の条件が成り立たなかった場合の処理を、「それ以外の場合は (else)」として付け足して書きたいときに使う構文です。

▼ 構文：if ～ else 文

```
if (条件) {
    処理1
} else {
    処理2
}
```



条件次第で
処理1か処理2を
実行する

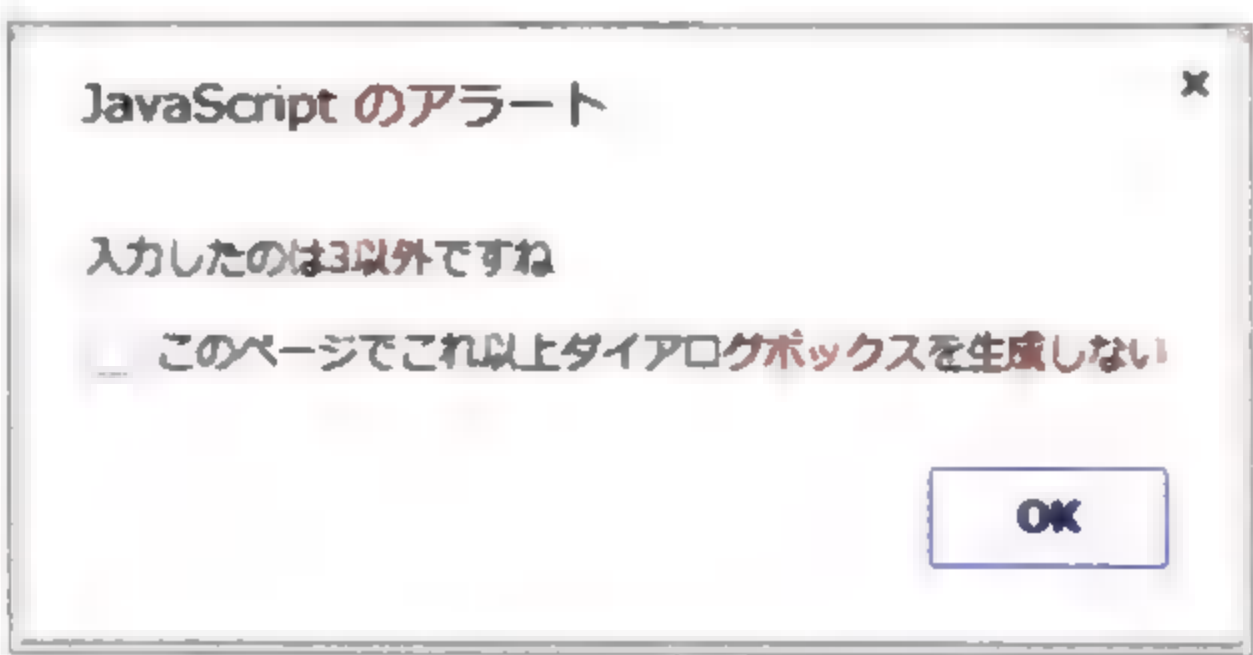
条件が成り立つとき (true のとき) は、処理1を実行します。逆に、条件が成り立たないとき (= false のとき) は、処理2を実行します。

● if ～ else 文のサンプルプログラム

サンプルプログラムを使って、実際にif ～ else 文の動作を確認してみましょう。以下のソースコードを入力して、実行してみてください。

```
1  var n = window.prompt(' 数値を 1 つ入力してください ');
2  if (n == 3) {
3      window.alert(' 入力した数値は 3 ですね '); // 処理 1
4  } else {
5      window.alert(' 入力したのは 3 以外ですね '); // 処理 2
6  }
```



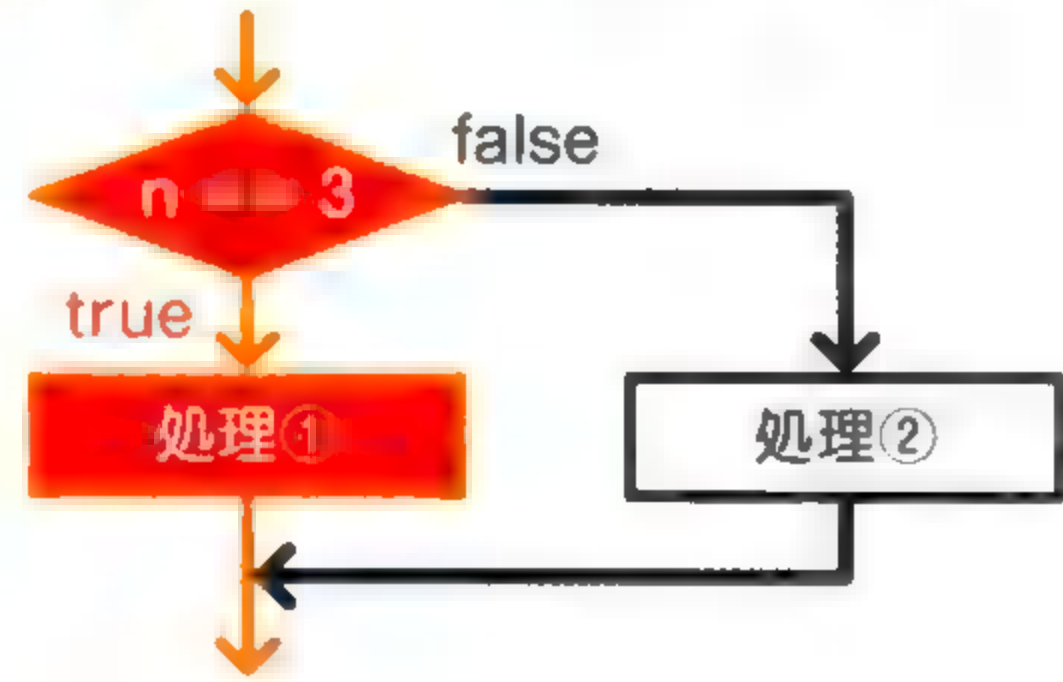


3以外を入力すると別のダイアログボックスが表示される

● サンプルプログラムの解説

まず、入力ダイアログボックスの入力欄に入力された数値が、変数nに代入されます (1行目)。条件は「n == 3」、すなわち「nは3と等しい」です (2行目)。変数nの値が3なら条件はtrueとなり、「入力した数値は3ですね」という警告ダイアログボックスが表示されて、プログラムは終了します (3行目)。

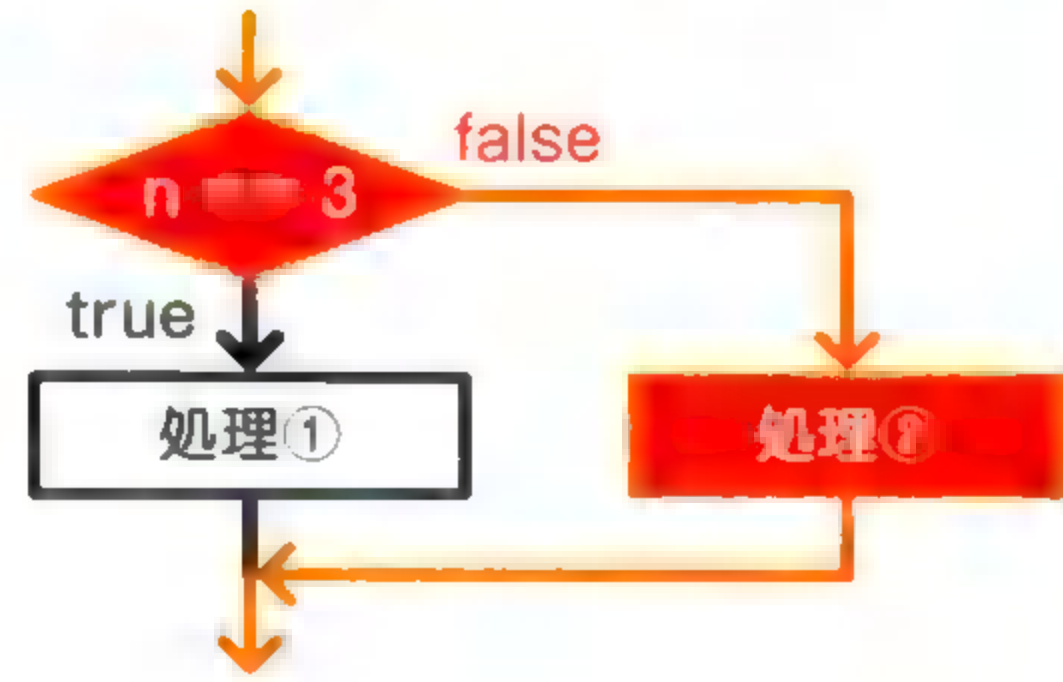
▼ 条件が true のときの処理の流れ



条件がtrueのときは処理1を実行する

もし変数nに3以外の値が入力されると、条件はfalseとなり、3行目をスキップし、else (4行目) 以下の処理である「入力したの3以外ですね」というダイアログボックスが表示されて (5行目)、プログラムは終了となります。

▼ 条件が false のときの処理の流れ



条件がfalseのときは処理2を実行する

Lesson 1

SECTION

3

条件文③—— if～else if～else 文

3つ目の条件文は、if～else if～else 文です。これまでに学んだ if else 文を複数組み合わせた条件文で、さらに細かく条件分岐をさせたいときに使います。

● if～else if～else 文の書き方

if～else if～else (イフ・エルスイフ・エルス) 文は、条件によって、さらに細かく実行したい処理を選択したいときに使う構文です。

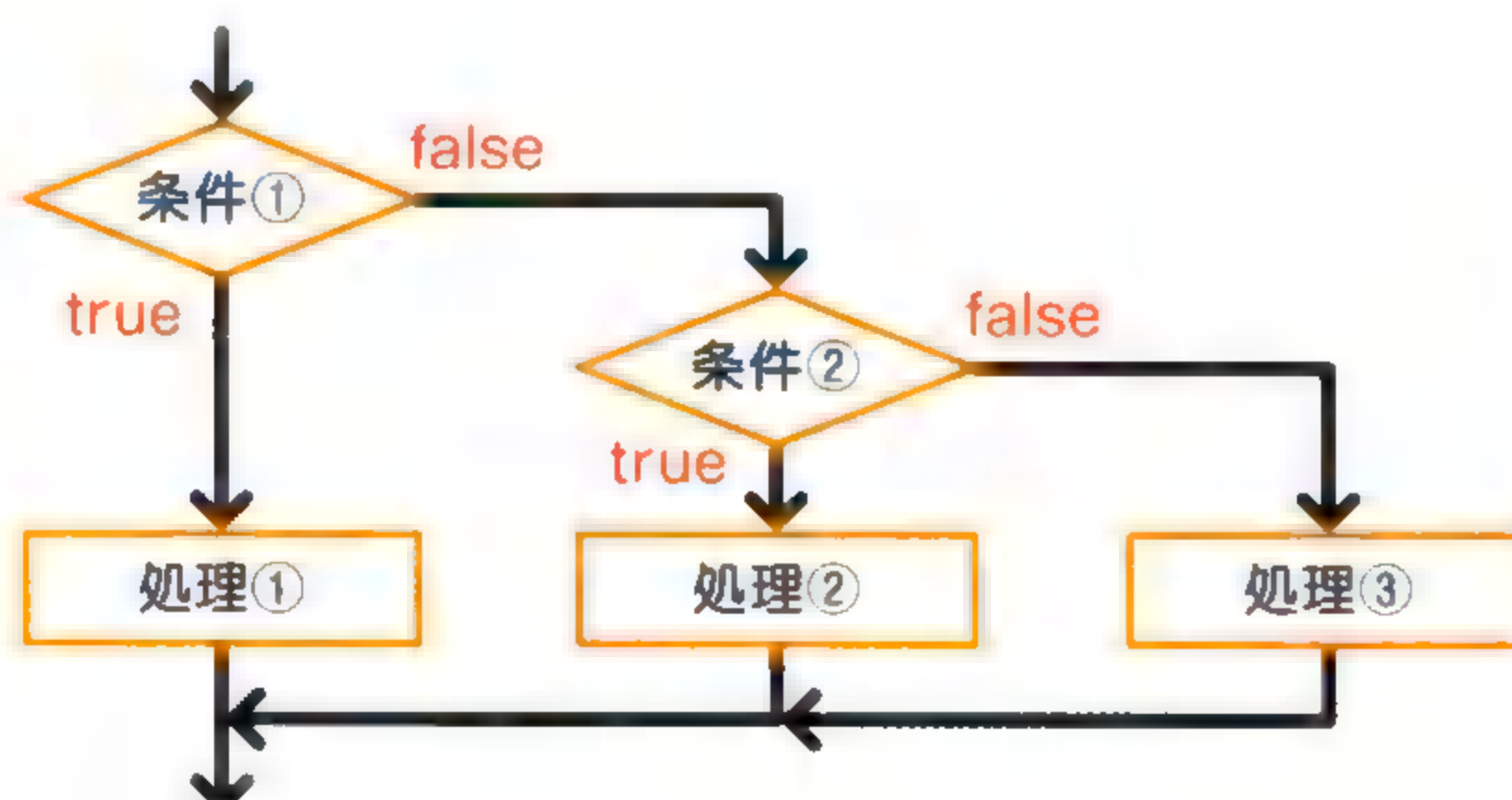
▼ 構文：if～else if～else 文

```
if (条件1) {  
    処理1  
} elseif (条件2) {  
    処理2  
} else {  
    処理3  
}
```

条件1がtrueのときは、処理1を実行します。

条件1はfalseだけれども、条件2がtrueのときは、処理2を実行します。

条件1も条件2もfalseのときは、処理3を実行します。



条件次第で
処理1か処理2か処理3を
実行する

● if ~ else if ~ else 文のサンプルプログラム

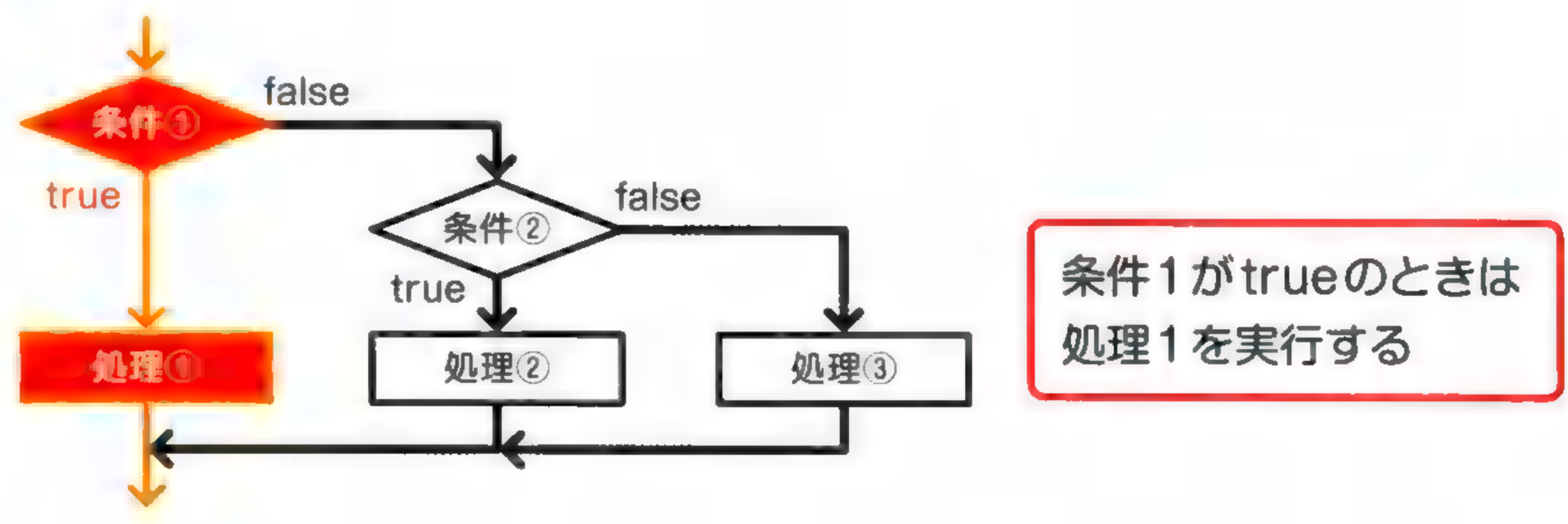
サンプルプログラムを使って、実際に if ~ else if ~ else 文の動作を確認してみましょう。以下のソースコードを入力して、実行してみてください。

```
1 var n = window.prompt(' 数値を 1 つ入力してください ');
2 if (n == 3) {
3     window.alert(' 入力した数値は 3 ですね '); // 処理 1
4 } else if (n > 3) {
5     window.alert(' 入力した数値は 3 より大きいですね '); // 処理 2
6 } else {
7     window.alert(' 入力したのは 3 より小さいですね '); // 処理 3
8 }
```

● サンプルプログラムの解説

まず、入力ダイアログボックスの入力欄に入力された数値が、変数 n に代入されます (1 行目)。条件は「n == 3」、すなわち「n は 3 と等しい」です (2 行目)。変数 n の値が 3 なら条件は true となり、「入力した数値は 3 ですね」という警告ダイアログボックスが表示されて、プログラムは終了します (3 行目)。

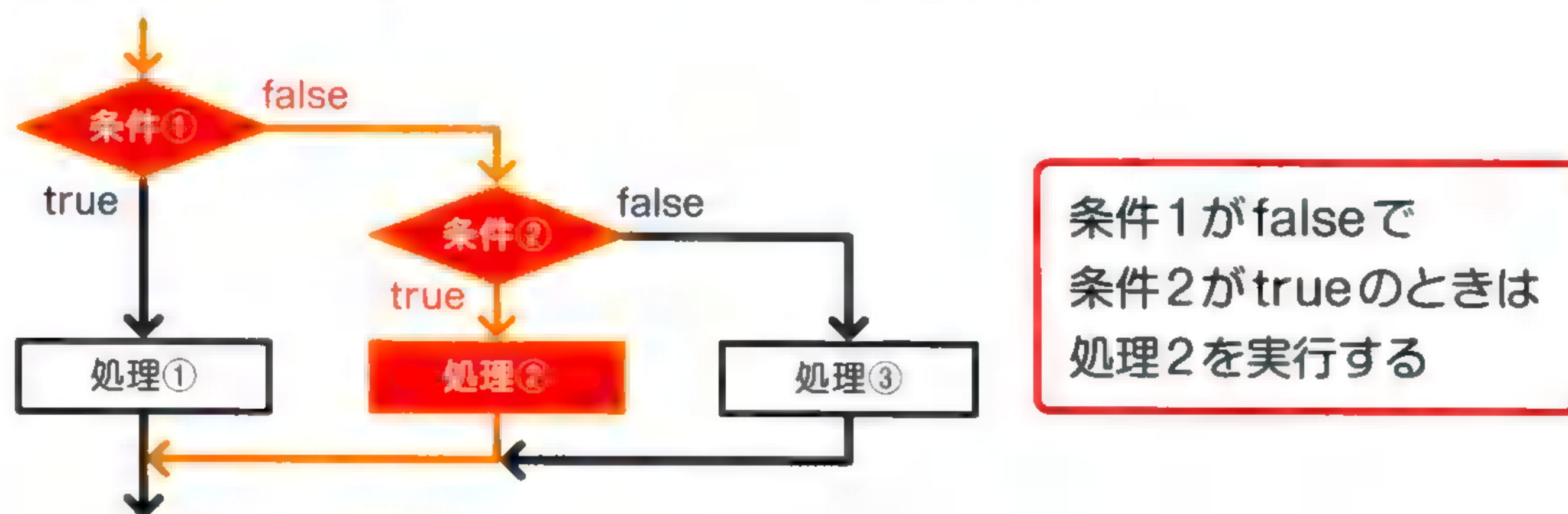
▼ 条件 1 が true のときの処理の流れ



もし変数 n に 3 以外の値が入力されると、条件 1 は false となり、3 行目をスキップし、条件 2 (4 行目) に行きます。

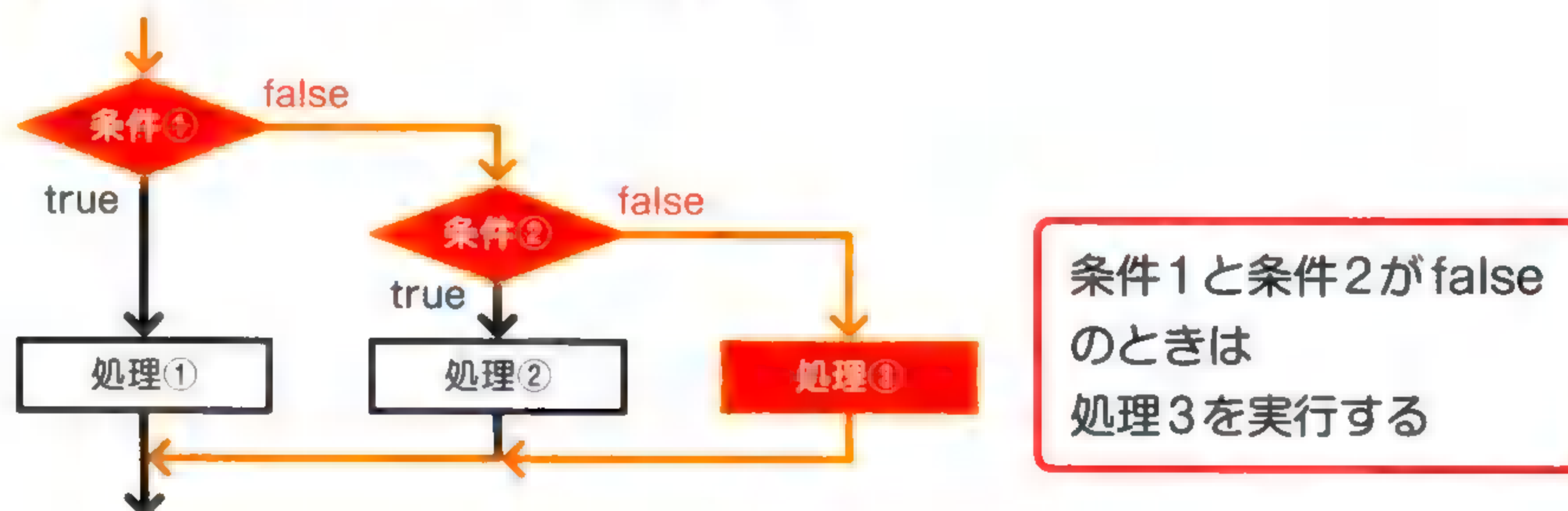
条件 2 は「n > 3」、すなわち「n は 3 より大きい」です (4 行目)。変数 n の値が 3 より大きいなら条件 2 は true となり、「入力した数値は 3 より大きいですね」という警告ダイアログボックスが表示されて、プログラムは終了します (5 行目)。

▼ 条件 1 が false で、条件 2 が true のときの処理の流れ



もし変数 n に 3 より小さい数値が入力されると、条件 1、条件 2 は false となり、3～5 行目をスキップして 6 行目に行きます。そして、else (6 行目) 以下の処理である「入力した数値は 3 より小さいですね」というダイアログボックスが表示されて (7 行目)、プログラムは終了となります。

▼ 条件 1、条件 2 が false のときの処理の流れ



● else if ～はいくつでも重ねられる

if ～ else if ～ else 文の「else if ～」は複数書くことも可能です。より細かく条件分岐することができます。たとえば、先のサンプルプログラムは、数値以外のデータが入力されても「入力したのは 3 より小さいですね」というダイアログが出てしまうのですが、「else if ～」をもう 1 つ使えば (6 行目)、数値以外のデータが入力された際に、「数値を入力してください」というダイアログが出るように修正することができます。

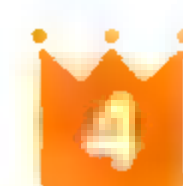
```
6   } else if (n < 3) {  
7       window.alert(' 入力した数値は 3 より小さいですね '); // 処理 3  
8   } else {  
9       window.alert(' 数値を入力してください '); // 処理 4  
10  }
```




条件文④

—— switch 文

最後の条件文は switch 文です。選択肢が多い場合の条件分岐によく使われます。



● switch 文の書き方

switch 文は、複数の選択肢の中から、条件に合う処理を選択して実行したいときに使う構文です。条件がとりうる値が多いときには、if ~ else if ~ else 文よりも簡潔に書くことができます。

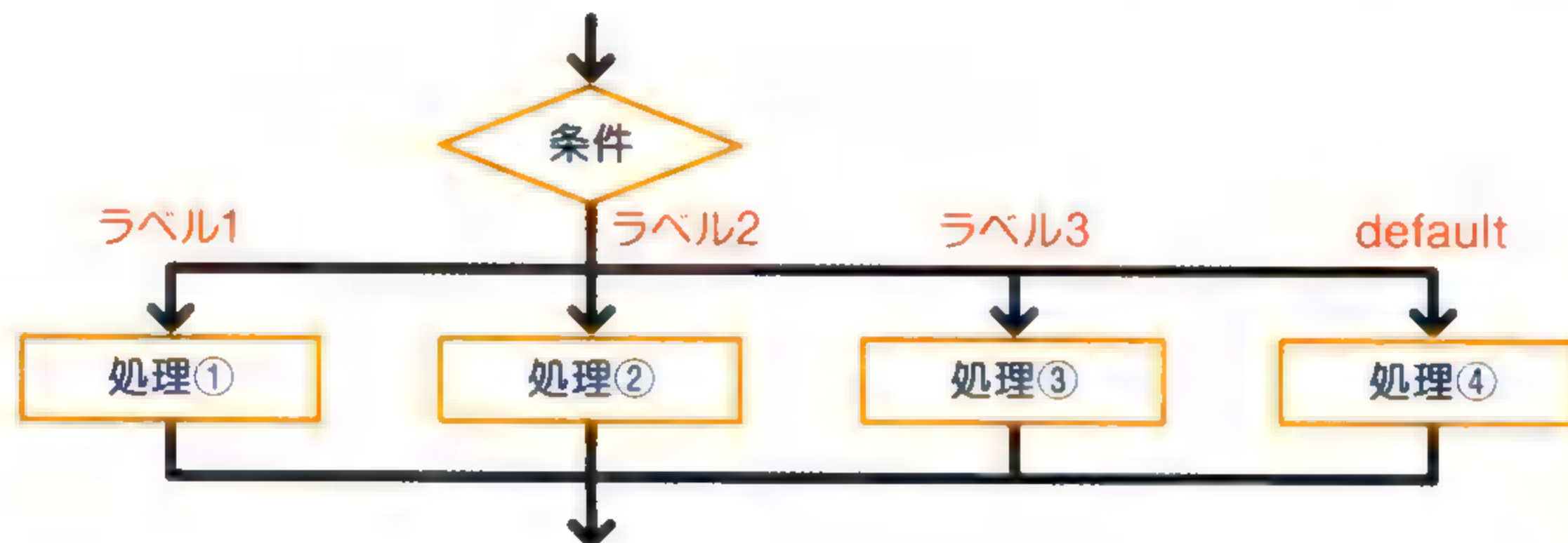
▼ 構文：switch 文

```
switch (条件) {  
    case ラベル1 :  
        処理1  
        break;  
    case ラベル2 :  
        処理2  
        break;  
    case ラベル3 :  
        処理3  
        break;  
    default :  
        処理4  
}
```

switch 文は、条件の値と、case のあとに書かれた値（ラベル）を、上から順番に比較していきます。条件の値に一致するラベルがあれば、その case のところにある処理を実行します。処理の中に break 文があれば、switch 文の実行を終了し、次の文に移ります。break 文がなければ、次の case 文が続けて実行されてしまい、狙ったような処理の選択になりません。

一致するラベルがないと、default文の処理を実行します。default文がなければ、switch文の実行を終了し、次の文に移ります。

caseのラベルと処理は、いくつでも書くことができます。



● switch 文のサンプルプログラム

サンプルプログラムを使って、実際にswitch文の動作を確認してみましょう。以下のソースコードを入力して、実行してみてください。

```
1  var n = window.prompt(' 名字を入力してください ');
2  switch (n) {
3      case ' 佐藤 ' :
4          window.alert(' 日本で一番多い名字ですね '); // 処理 1
5          break;
6      case ' 鈴木 ' :
7          window.alert(' 佐藤さんの次に多い名字ですね '); // 処理 2
8          break;
9      case ' 高橋 ' :
10         window.alert(' ベスト 3 にランクインしてますね '); // 処理 3
11         break;
12     default :
13         window.alert(' けっこう珍しい名字ですね '); // 処理 4
14 }
```

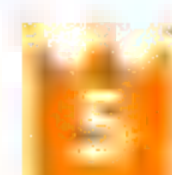
● サンプルプログラムの解説

変数nに入力された値が「佐藤」なら処理1が実行されます。「鈴木」なら処理2が、「高橋」なら処理3が実行されます。それら以外の値が入力された場合は、defaultの処理4が実行され、switch文は終了します。



条件の書き方

条件文の条件である条件式を書くには、比較演算子や論理演算子を使います。



● 条件式とは？

条件文や繰り返し文の条件を書くのに使う式のことを、**条件式**といいます。すでにサンプルプログラムで使った「`n == 3`」や「`n > 3`」や「`n < 3`」が条件式です。この条件式の中の「`==`」「`>`」「`<`」といった記号のことを**比較演算子**といいます。

● 比較演算子

比較演算子には、以下のような種類があります。

演算子	名前	条件式が true になる場合
<code><</code>	大なり	右辺の値が左辺の値より 大きい とき
<code><=</code>	大なりイコール	右辺の値が左辺の値 以上 のとき
<code>></code>	小なり	右辺の値が左辺の値より 小さい とき
<code>>=</code>	小なりイコール	右辺の値が左辺の値 以下 のとき
<code>==</code>	等値演算子	左辺と右辺の値が 等しい とき
<code>!=</code>	不等値演算子	左辺の値と右辺の値が 等しくない とき
<code>===</code>	厳密等価演算子	左辺と右辺の、 値とデータ型が等しい とき
<code>!==</code>	厳密不等価演算子	左辺と右辺の、少なくとも 値かデータ型が等しくない とき

「`<`」を使った条件式では、右辺の値が左辺の値より大きいとき、条件式は true になり、等しいか小さいときには false になります。

「`<=`」を使った条件式では、右辺の値が左辺の値より大きい、あるいは等しいとき、条件式は true になり、小さいときには false になります。

「`>`」を使った条件式では、右辺の値が左辺の値より小さいとき、条件式は true になり、等しいか大きいときには false になります。

「>=」を使った条件式では、右辺の値が左辺の値より小さいか、あるいは等しいとき、条件式はtrueになり、大きいときにはfalseになります。

「==」を使った条件式では、左辺と右辺の値が等しいとき、条件式はtrueになり、等しくないときはfalseになります。たとえば、文字列の「3」と数値の「3」は、データ型は異なりますが値は等しいと見なされるのでtrueになります。

「==」の反対が「!=」です。「!=」を使った条件式では、左辺の値と右辺の値が等しくなければ、条件式はtrueになり、等しければfalseになります。たとえば、文字列の「3」と数値の「3」は値が等しいと見なされ、falseになります。

「===」を使った条件式では、左辺と右辺の値とデータ型がどちらも等しいとき、条件式はtrueになり、どちらかが等しくないときはfalseになります。たとえば、文字列の「3」と数値の「3」は、データ型が異なるのでfalseになります。

「===」の反対が「!==」です。「!==」を使った条件式では、左辺と右辺の値が等しくないか、あるいはデータ型が等しくなければtrueになり、値とデータ型が両方とも等しければfalseになります。

● 論理演算子

論理演算子は、より複雑な条件式を書くときに使う演算子です。比較演算子で作った条件式と組み合わせて使います。

演算子	名前	条件式が true になる場合	使用例
&&	論理積演算子	左右の式が両方とも true のとき	(n > 3) && (n < 10)
	論理和演算子	左右の式のどちらかが true のとき	(n < 3) (n > 10)
!	論理否定演算子	元の式が false のとき	!(n == 3)

● 比較演算子以外を使った条件式

条件文や繰り返し文の条件には、比較演算子や論理演算子を使った条件式だけでなく、文字列や数値を使うこともできます。文字列や数値などを使った場合、trueとfalseは以下のように判定されます。

	true になる場合	false になる場合
数値	0 以外の数値	0 (数値)、NaN (0 を 0 で割った値など)
文字列	1 文字以上の文字列	空文字 (" や "" など 0 文字の文字列)
その他		null、undefined





Day 2 Lesson 2

繰り返し文

このレッスンでは、同じ処理を繰り返したいときに使う繰り返し文のしくみと使い方を学びます。

- 1 繰り返し文 ① — while 文
- 2 繰り返し文 ② — for 文
- 3 繰り返し文 ③ — do ~ while 文
- 4 for 文と配列を使った繰り返し処理
- 5 インクリメント演算子と
デクリメント演算子

繰り返し文①

while 文



1つ目の繰り返し文は while 文です。while は英語で「～のあいだ」という意味ですが、JavaScript でも同じく、「ある条件が成り立っているあいだ」という意味合いで使われています。

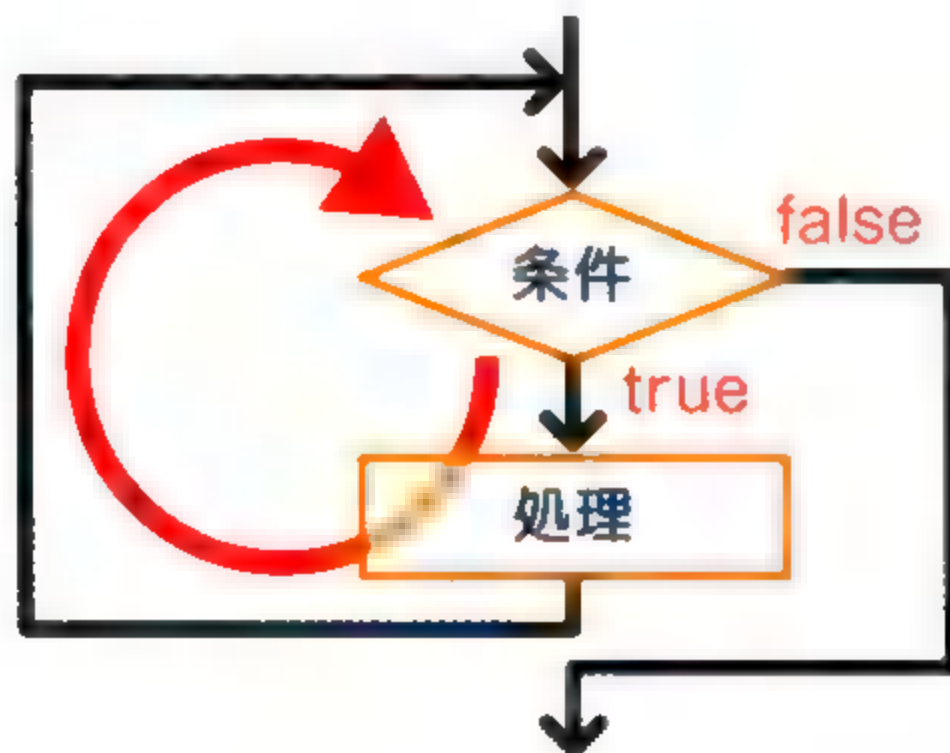
● while 文の書き方

while 文は、同じ処理を繰り返し実行したいときに使う構文です。プログラミング用語では、繰り返しのことを「ループ」と言ったりもしますので、while ループと呼ばれることもあります。

▼ 構文：while 文

```
while (条件) {
  処理
}
```

条件が成り立っているあいだは、何度も処理が繰り返されます。



条件がtrueのあいだは
何度も処理を実行する

● while 文を使ったサンプルプログラム

実際に while 文を使ったサンプルプログラムを作ってみましょう。変数*i*の値を、0から2まで、1ずつ増やしながら、ブラウザの画面に出力するプログラムです。ソースコードを入力して、実行してみてください。

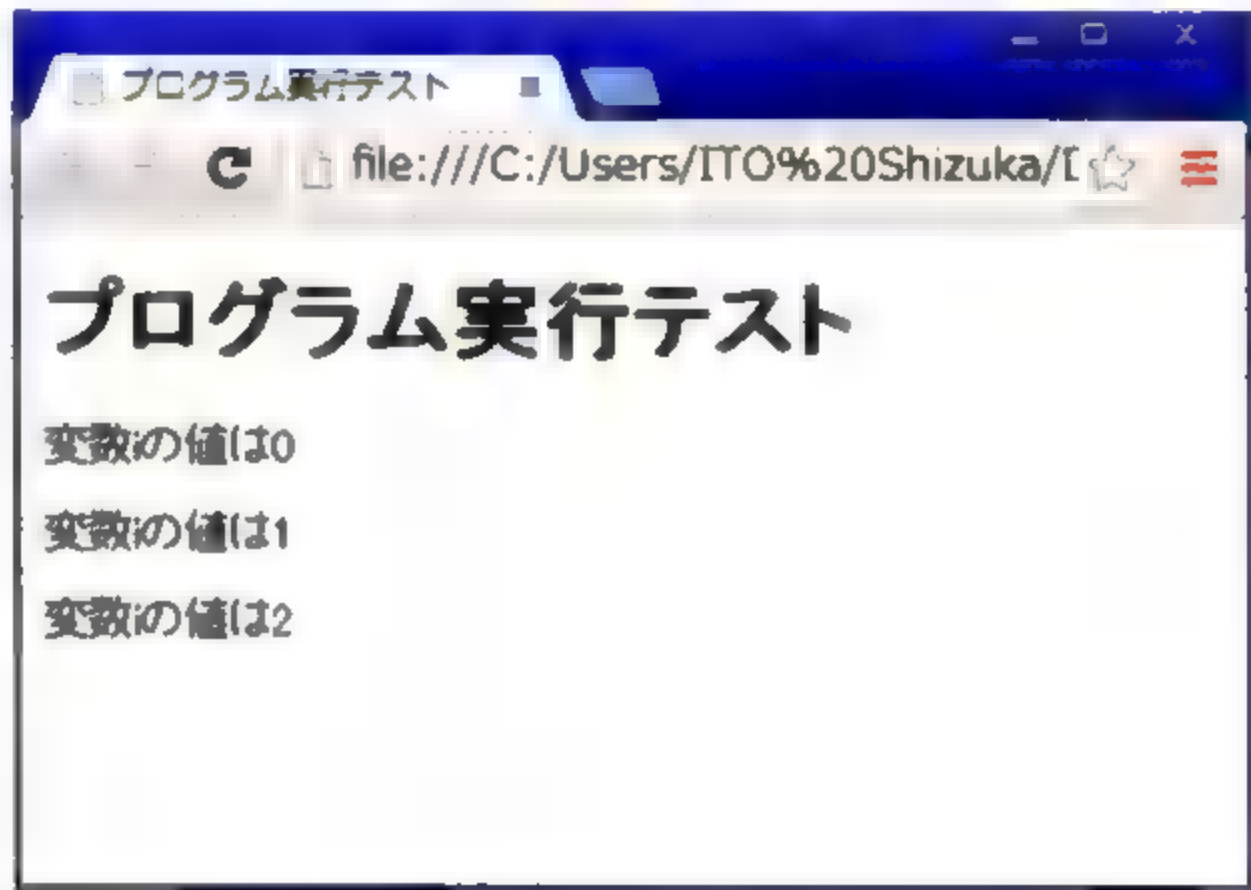



```

1  var i = 0;
2  while (i < 3) {
3      document.writeln('<p>変数 i の値は '+ i + '</p>');
4      i++;
5  }

```

プログラムを実行すると、以下のような画面が表示されます。



文字列が3行表示される

MEMO

document.writeln() は、ブラウザ画面に文字を出力するメソッドです。3-1-8で説明します。

● サンプルプログラムの解説

while 文の前で、変数 i を宣言しています。初期値は 0 (数値) です (1 行目)。

2 行目から 5 行目が while 文になります。while 文の条件は「i < 3」です。つまり「i が 3 より小さい」あいだは、条件は true になります (2 行目)。

while 文の処理ブロックには 2 つの文が含まれています。

1 つめの文は、画面にその時点の**変数 i の値を表示する文**です。

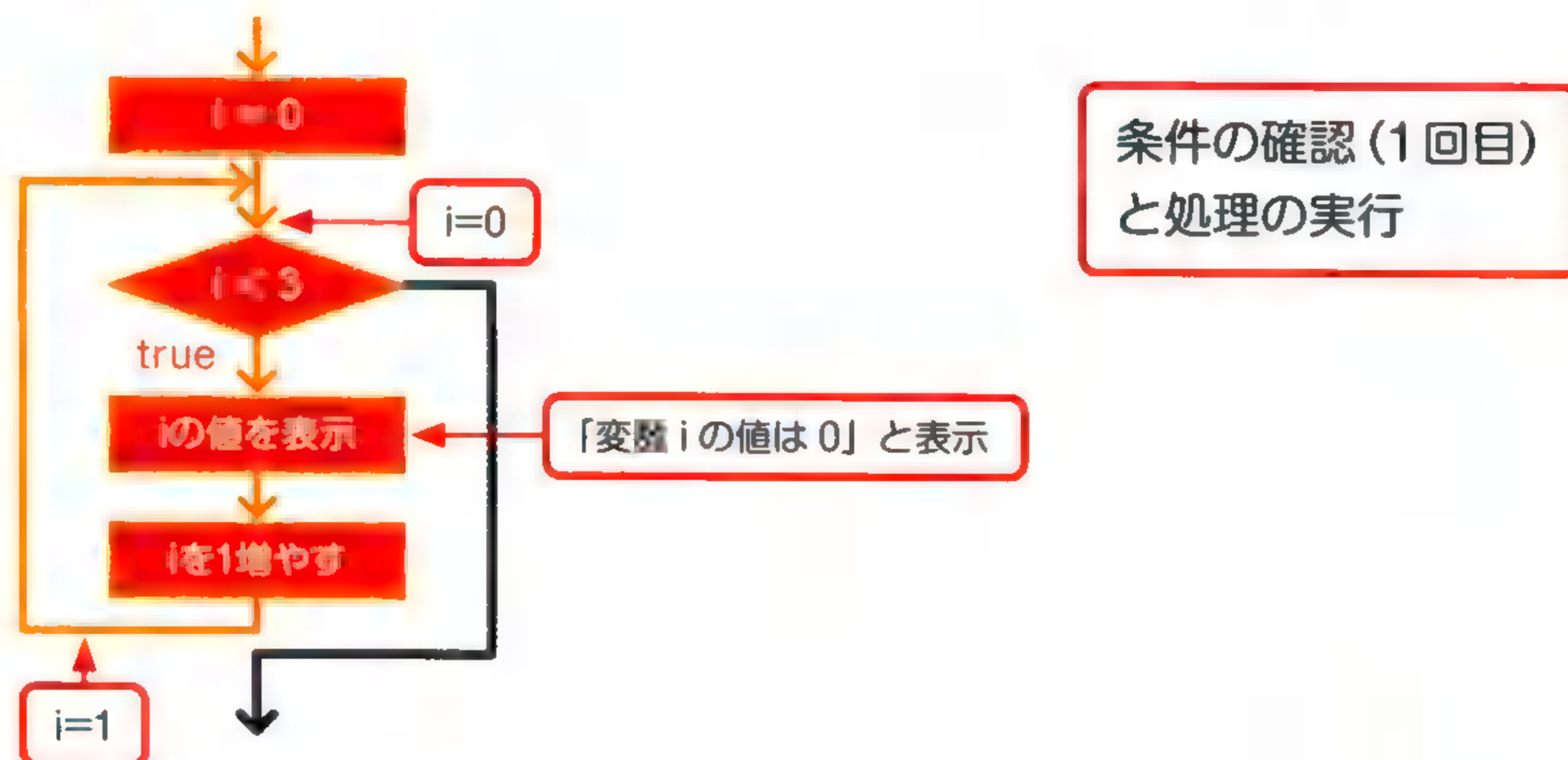
2 つめの文は、**i の値を 1 つ増やす文**です (「++」の意味と使い方は 2-2-5 で説明します)。

このプログラムがどのように動くかをもう少し詳しく見てみましょう。

MEMO

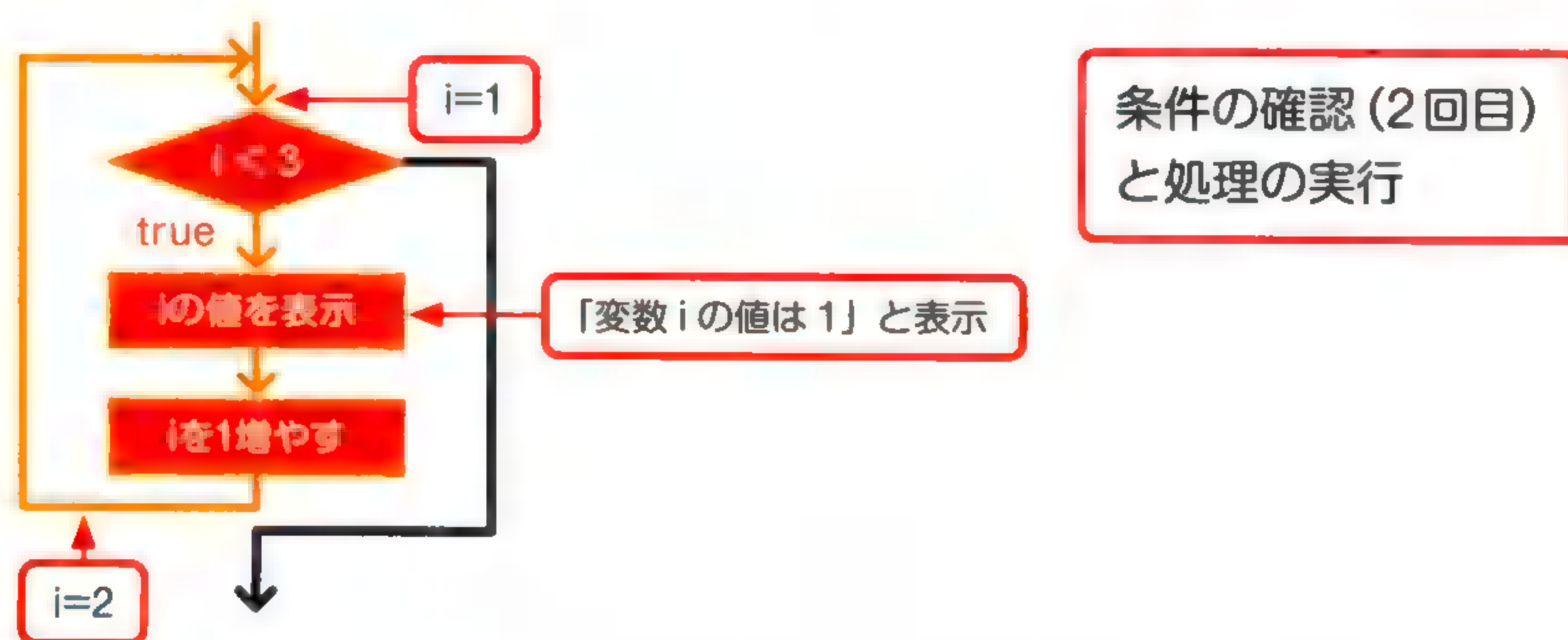
この変数 i のことを、**カウンタ** (カウンタ変数) と呼びます。繰り返しの回数をカウントする役割を果たしているからです。

[1回目の確認] 変数*i*の初期値は0の状態、while文の条件の確認に入ります。



while文の条件は「変数*i*が3より小さい」です。この時点では、**変数*i*は0**なので、条件は**true**、よって以下の処理 (3行目と4行目) が実行されます。「変数*i*の値は0」と画面に表示され (3行目)、*i*の値が1増えて1になり (4行目)、再び条件の確認 (2行目) に戻ります。

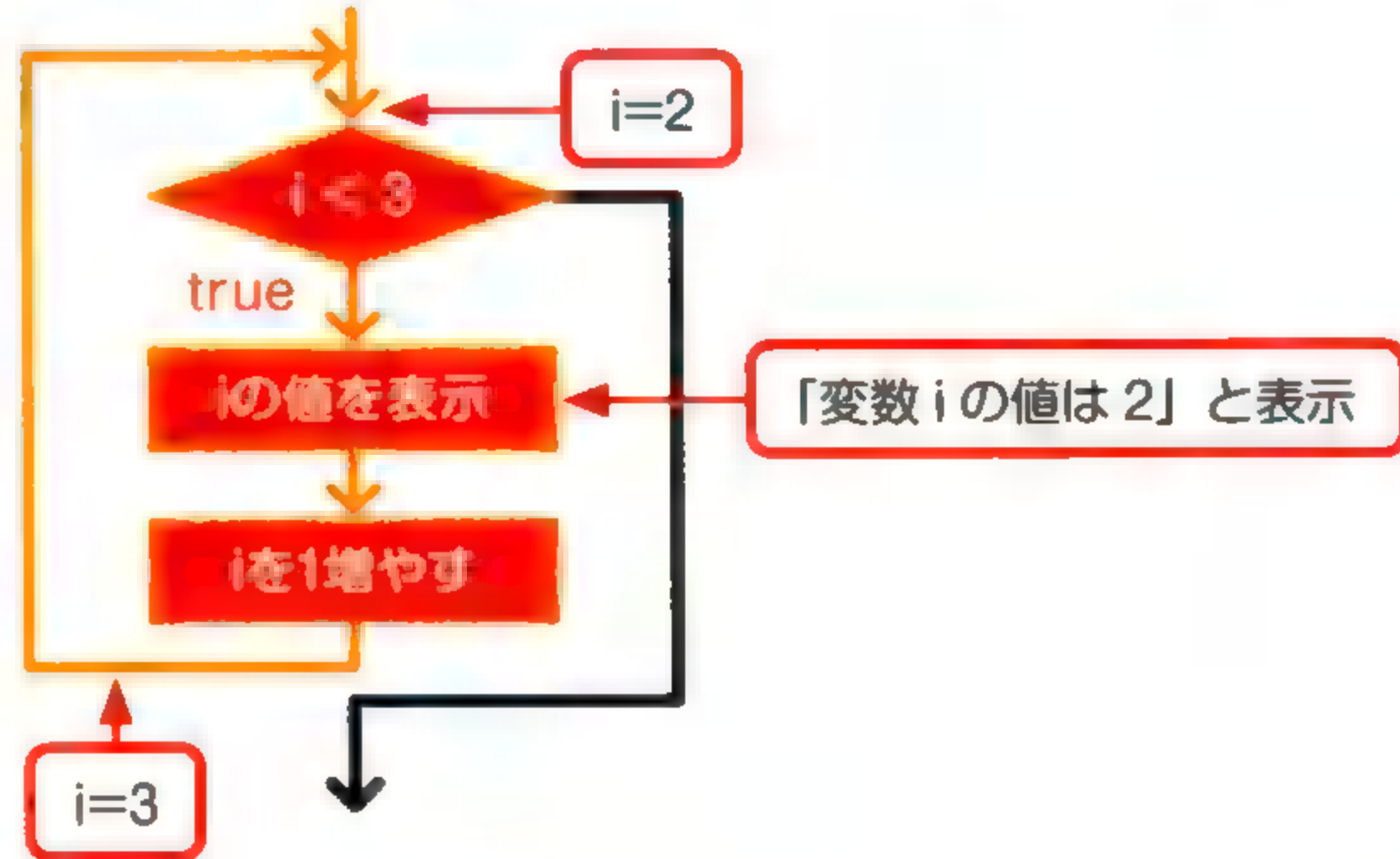
[2回目の確認] 変数*i*の値が1の状態、while文の条件の確認に入ります。



この時点の**変数*i*の値は1**ですので、条件は**true**です。よって処理が実行されます。「変数*i*の値は1」と画面に表示され、*i*の値が1増えて2になってから、再び条件に戻ります。



[3回目の確認] 変数*i*の値が2の状態、while文の条件の確認に入ります。

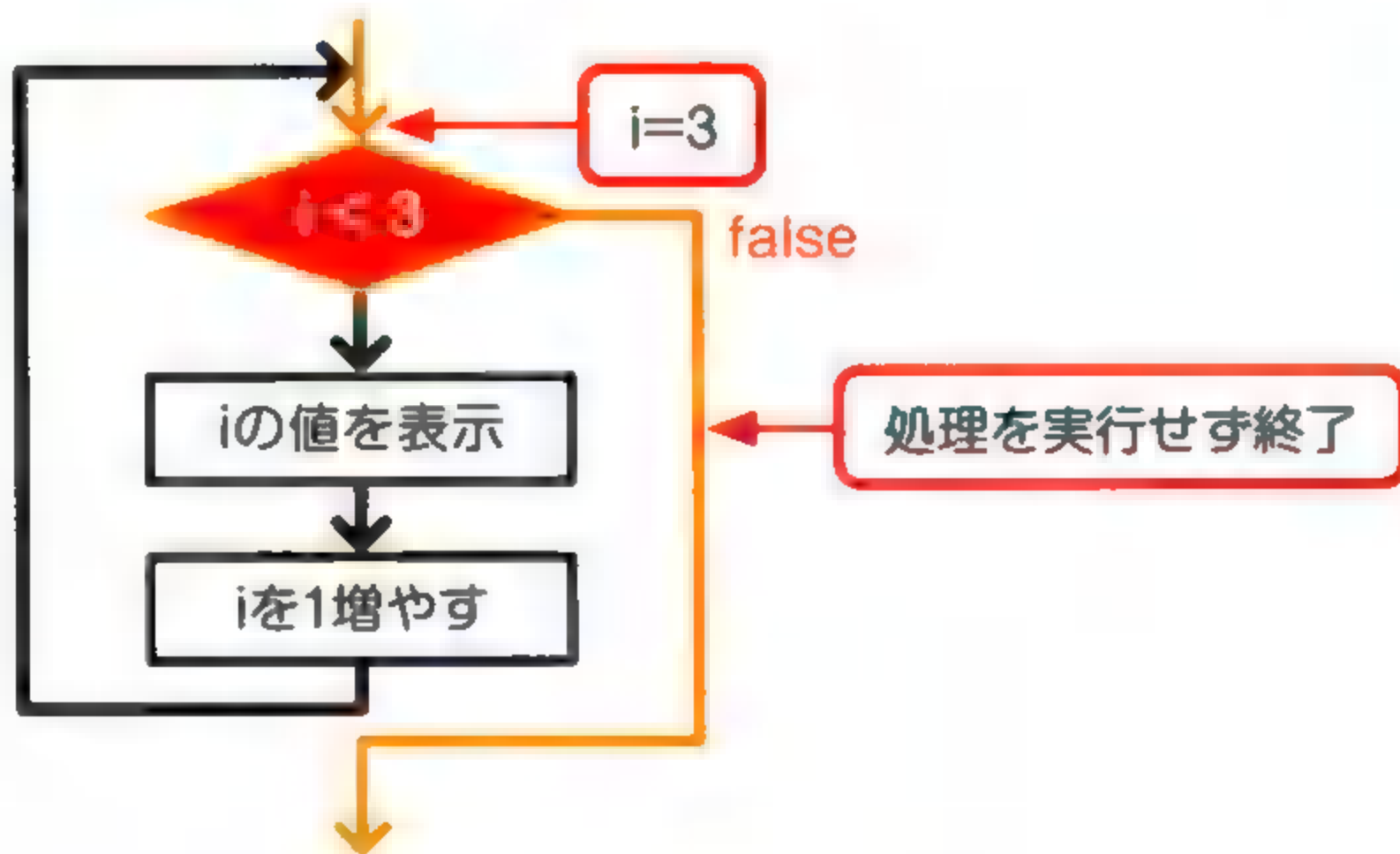


条件の確認 (3回目)
と処理の実行

「変数*i*の値は 2」 と表示

この時点の**変数*i*の値は2**ですので、条件は**true**です。よって処理が実行されます。「変数*i*の値は2」と画面に表示され、*i*の値が1増えて3になってから、再び条件に戻ります。

[4回目] 変数*i*の値が3の状態、while文の条件の確認に入ります。



条件の確認 (4回目)

処理を実行せず終了

この時点の**変数*i*の値は3**です。ここで異変が起こります。**条件「*i*が3より小さい」が成り立たない (false)** のです。条件がfalseになったので次の処理は実行されず、スキップされて、while文は終了します。

COLUMN

無限ループ

繰り返し文では条件の設定がもっとも重要なのですが、それと同じくらい大事なのが、4行目の「*i*++;」です。この文は変数*i*の値を1つずつ増やす役割をになっています。この文がないと、変数*i*は0のままなので、while文は永遠に実行され続け、画面には果てしなく「変数*i*の値は0」という文が表示されるでしょう。このような状態を**無限ループ**といいます。

Chromeで無限ループのプログラムを実行すると、ブラウザがフリーズ状態になります。フリーズしたら、ブラウザをいったん終了してください。

繰り返し文を使うときには、目的の回数実行したらプログラムが終了するような条件になっているか、確認しましょう。

Lesson 2

繰り返し文② ——for 文

SECTION

2

2つ目の繰り返し文は for 文です。for も英語では「～のあいだ」という意味です。while 文と同じく、「ある条件が成り立っているあいだ」繰り返す文です。

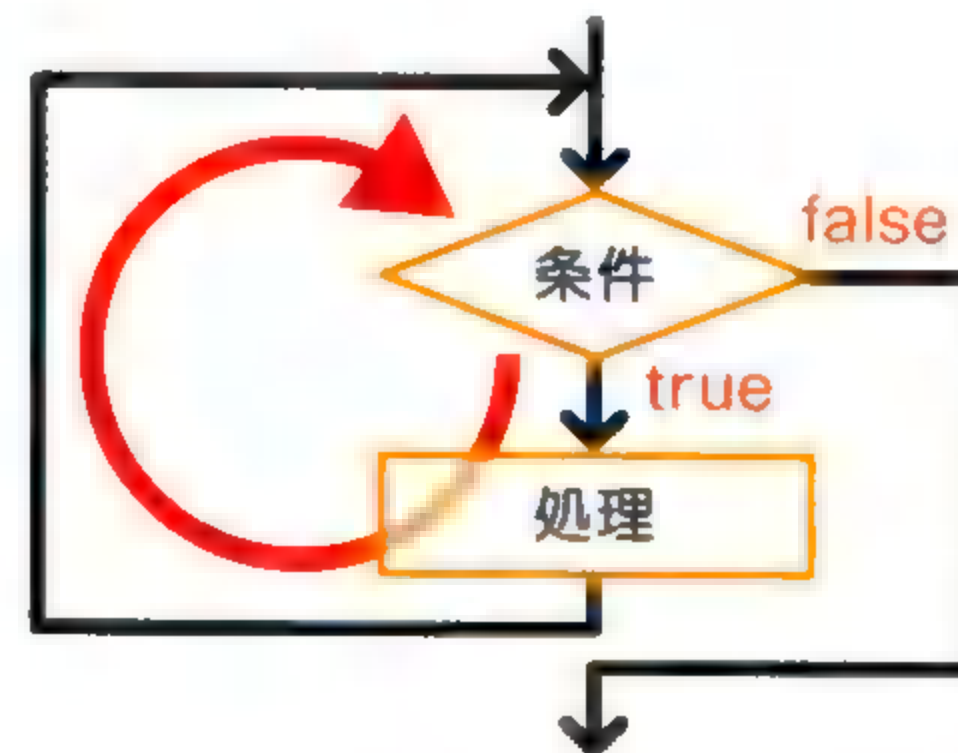
● for 文の書き方

for 文も、同じ処理を繰り返し実行したいときに使う構文です。for ループと呼ばれることもあります。while 文と違うのは、条件の部分で「カウンタ初期値の設定」「条件」「カウンタ値の更新」の3つを同時に指定できるようになっているところです。

▼ 構文：for 文

```
for ( 初期値 ; 条件 ; 更新 ) {  
    処理  
}
```

条件が成り立っているあいだは、何度も処理が繰り返されます。



条件がtrueのあいだは
何度でも処理を実行する

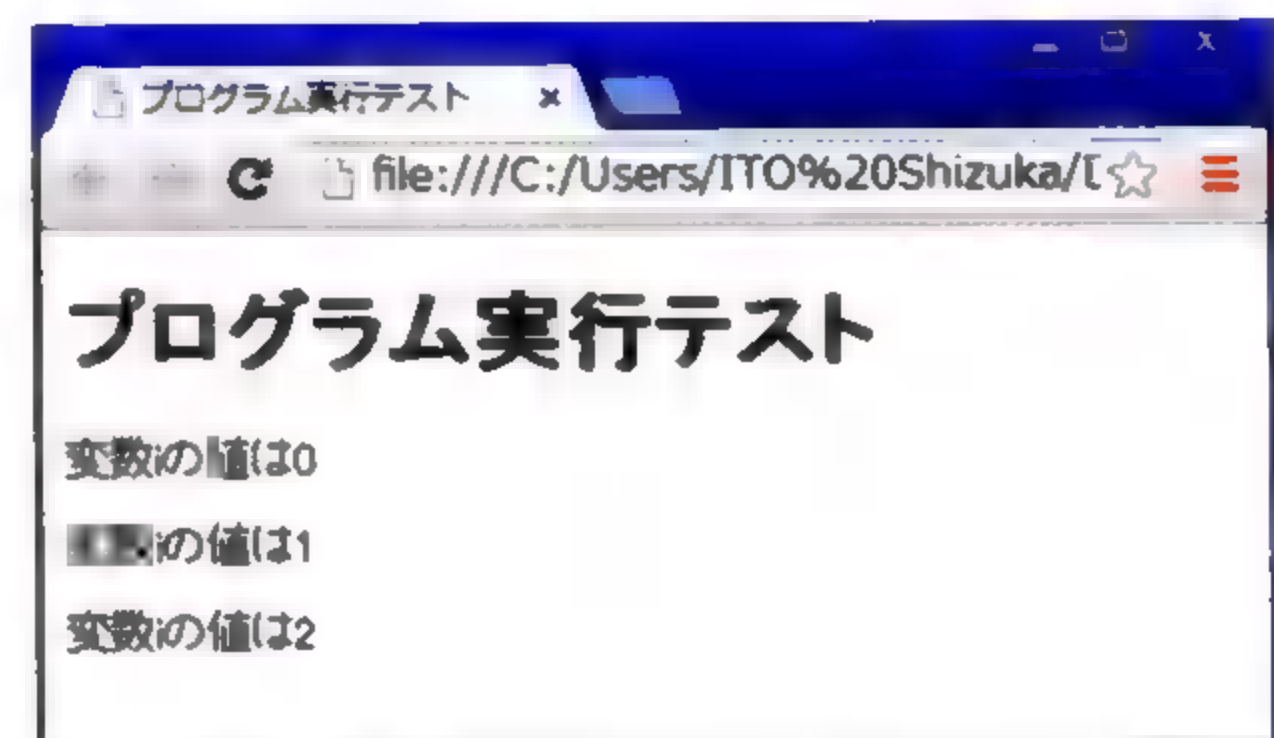
● for 文を使ったサンプルプログラム

では、for 文を使ったサンプルプログラムを作ってみましょう。変数*i*の値を、0から2まで、1ずつ増やしながら、ブラウザの画面に出力するプログラムです。ソースコードを入力して、実行してみてください。

```
1  for (var i = 0; i < 3; i++) {  
2      document.writeln('<p> 変数の値は ' + i + '</p>');  
3  }
```



プログラムを実行すると、以下のような画面が表示されます。



文字列が3行表示される

MEMO

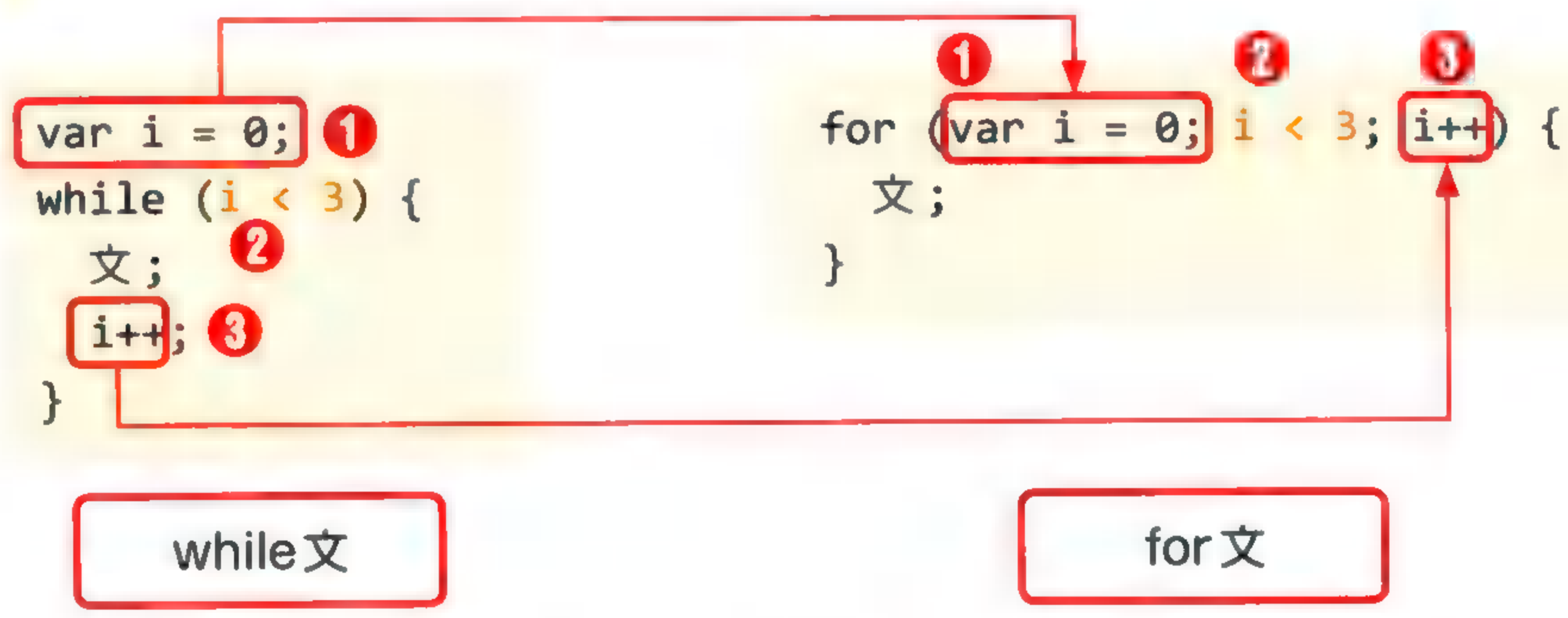
for文の「i++」のあとにはセミコロン (;) は付けません。付けるとエラーになります。

● while 文と for 文の違い

このプログラムの動作自体は、前のセクションで解説したサンプルプログラムとまったく同じですので、前のセクションを見ていただくとして、代わりに、while文とfor文の違いと、for文の注意点を説明しておきたいと思います。

for文では、while文の前にあった「var i = 0;」と4行目の「i++」が、for文のカッコの中に移動してきています。

▼ while 文から for 文へ



for文では「i++」が処理ブロックの前に来ていますが、iの値が1増えるのは処理ブロックが実行されたあと（while文と同じ）なので、この点勘違いしないようにしましょう。

while文とfor文、どちらを使うかですが、動作も実行結果も同じですので、好きな方を使っていただいてもかまいません。繰り返しの回数を明確に指定したい場合は、for文を使うといいでしょう。たとえば5回繰り返したいときは「i = 0; i < 5; i++」、30回繰り返したいときは「i = 0; i < 30; i++」などと書きます。

繰り返し文③

— do ~ while 文

SECTION

3

3つ目の繰り返し文は do ~ while 文です。名前が while 文と似ていますが、動作は異なります。間違えないように、違いを意識して覚えましょう。

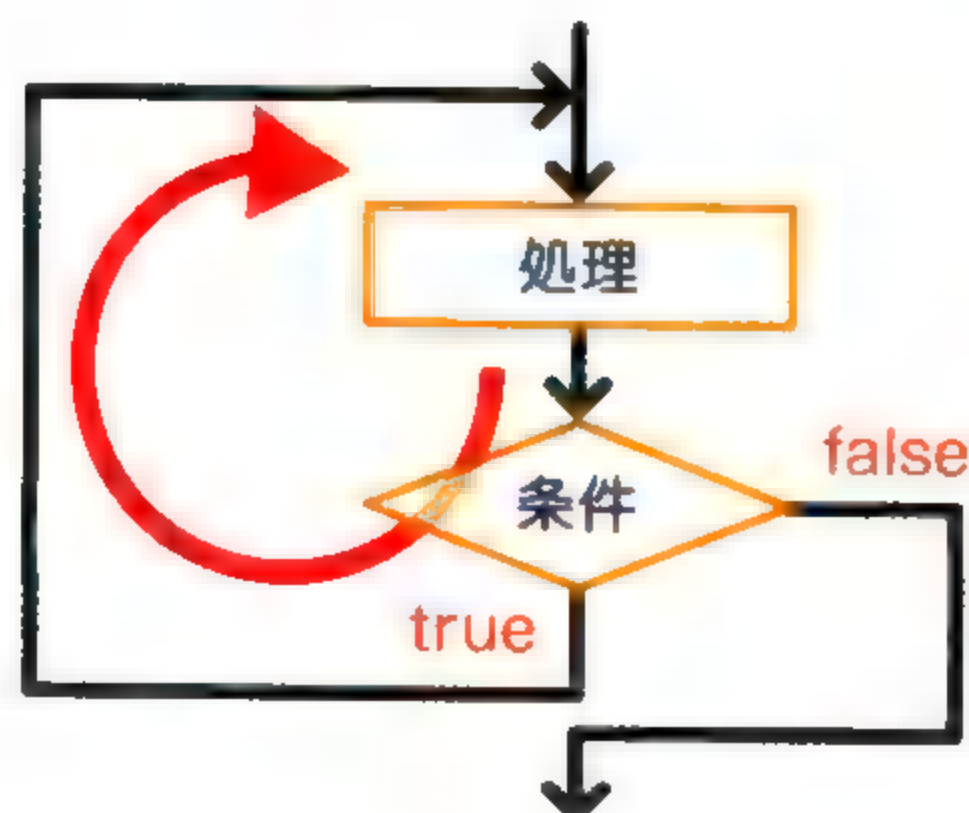
● do ~ while 文の書き方

do ~ while 文も、同じ処理を繰り返し実行したいときに使う構文です。do ~ while ループとも呼ばれます。while 文ではブロックの前にあった「while (処理)」がブロックの後ろに来ています。

▼ 構文：do ~ while 文

```
do {
    処理
} while (条件)
```

while 文と同じく、条件が成り立っているあいだは、何度も処理が繰り返されます。



条件が true のあいだは
何度でも処理を実行する

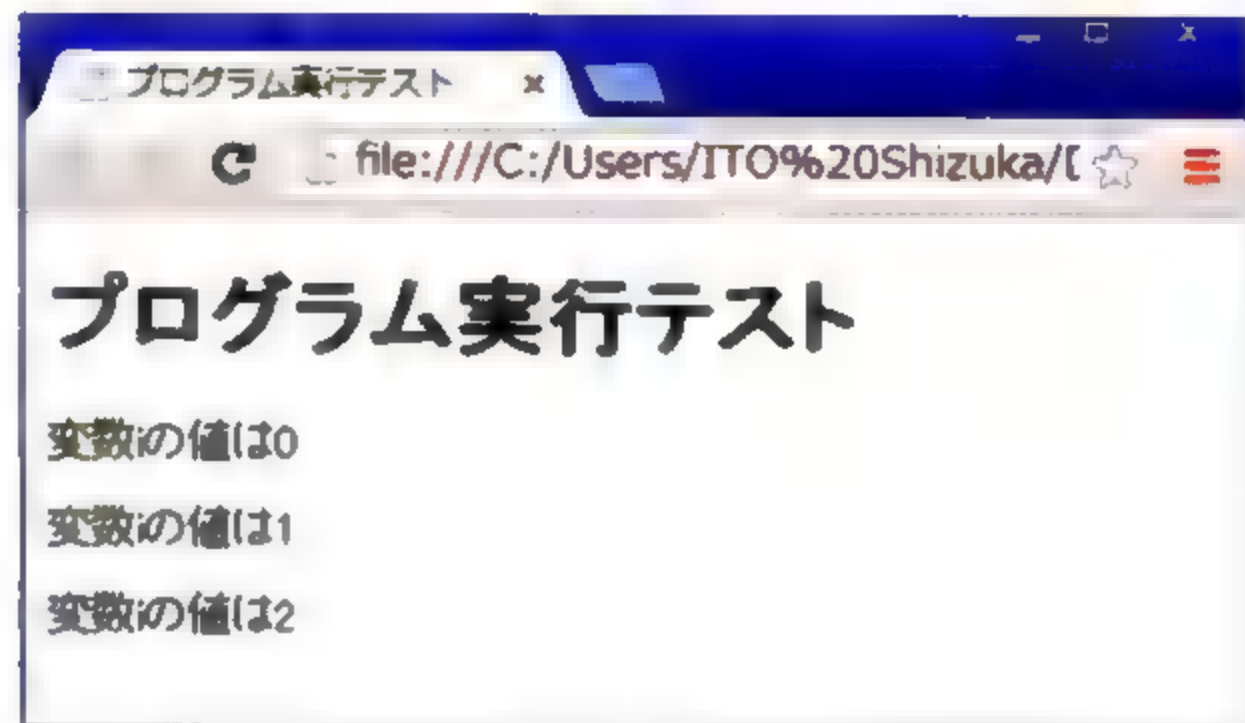
● do ~ while 文を使ったサンプルプログラム

では、do ~ while 文を使ったサンプルプログラムを作ってみましょう。変数 i の値を、0 から 2 まで、1 ずつ増やしながら、ブラウザの画面に出力するプログラムです。ソースコードを入力して、実行してみてください。




```
1 var i = 0;
2 do {
3   document.writeln('<p>変数 i の値は ' + i + '</p>');
4   i++;
5 } while (i < 3)
```

プログラムを実行すると、以下のような画面が表示されます。

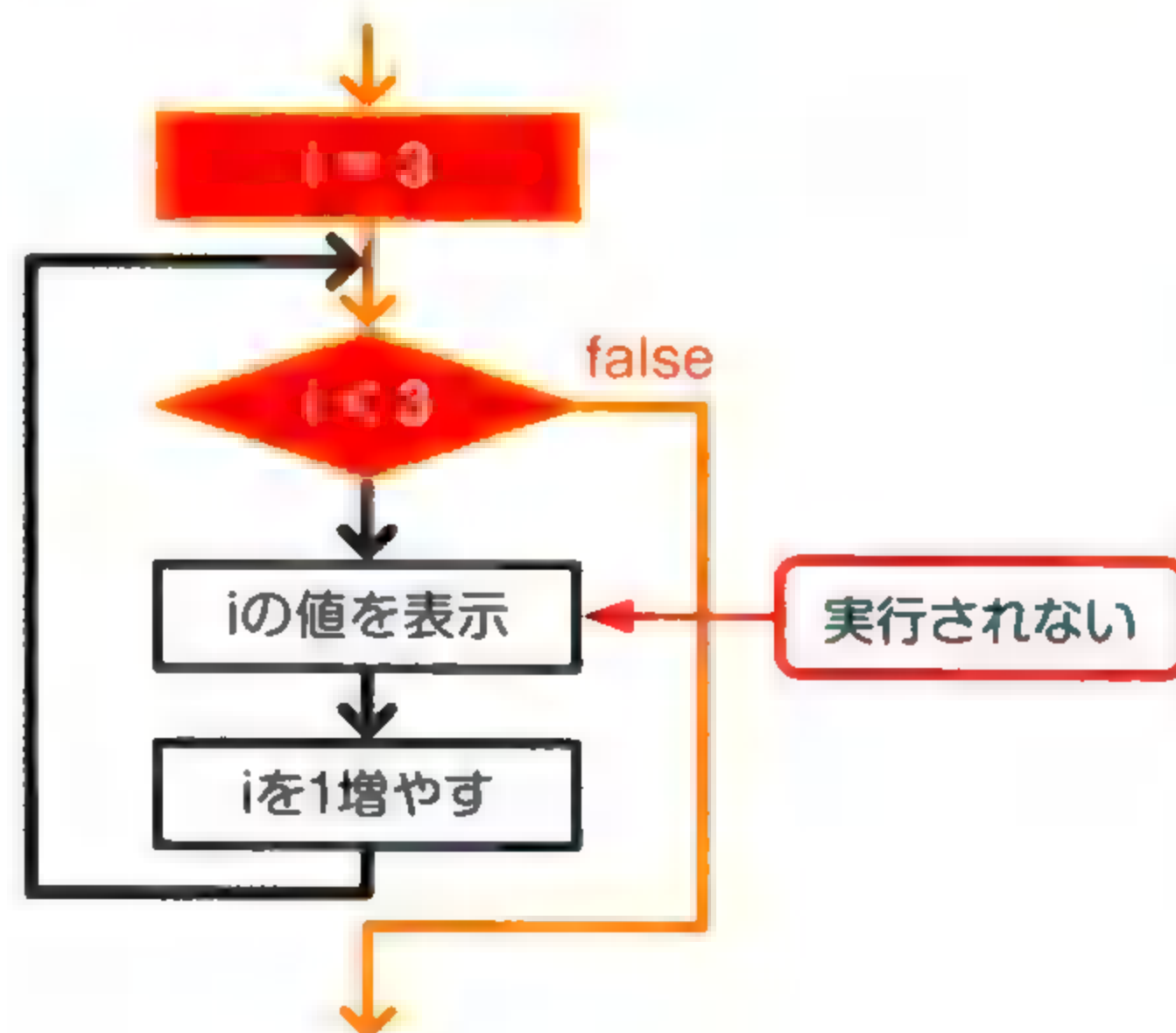


文字列が3行表示される

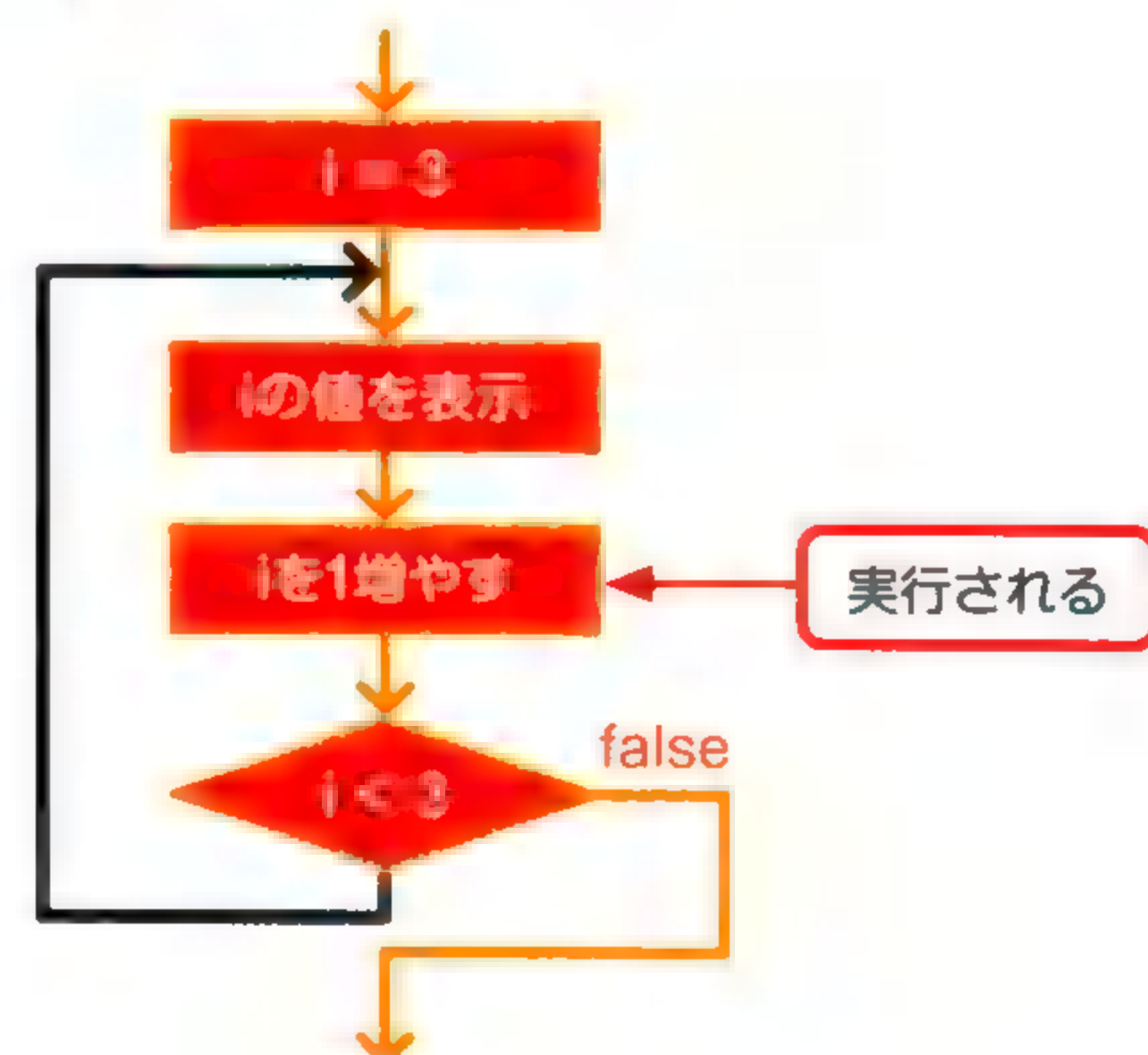
● while 文と do ~ while 文の違い

このプログラムの動作自体は、while 文や for 文のセクションで解説したサンプルプログラムとまったく同じですので、前のセクションを見ていただくとして、ここでは、while 文と do ~ while 文の違いを説明しておきます。

▼ while 文



▼ do ~ while 文



do ~ while 文は、条件の判定がブロックの後ろに来ているため、条件が成り立たない場合でも、**最低1回は処理が実行**されます。これに対して、while 文は、条件判定が前にあるため、条件が成り立たない場合は、処理は1回も実行されずに終わります。

for 文と配列を使った 繰り返し処理

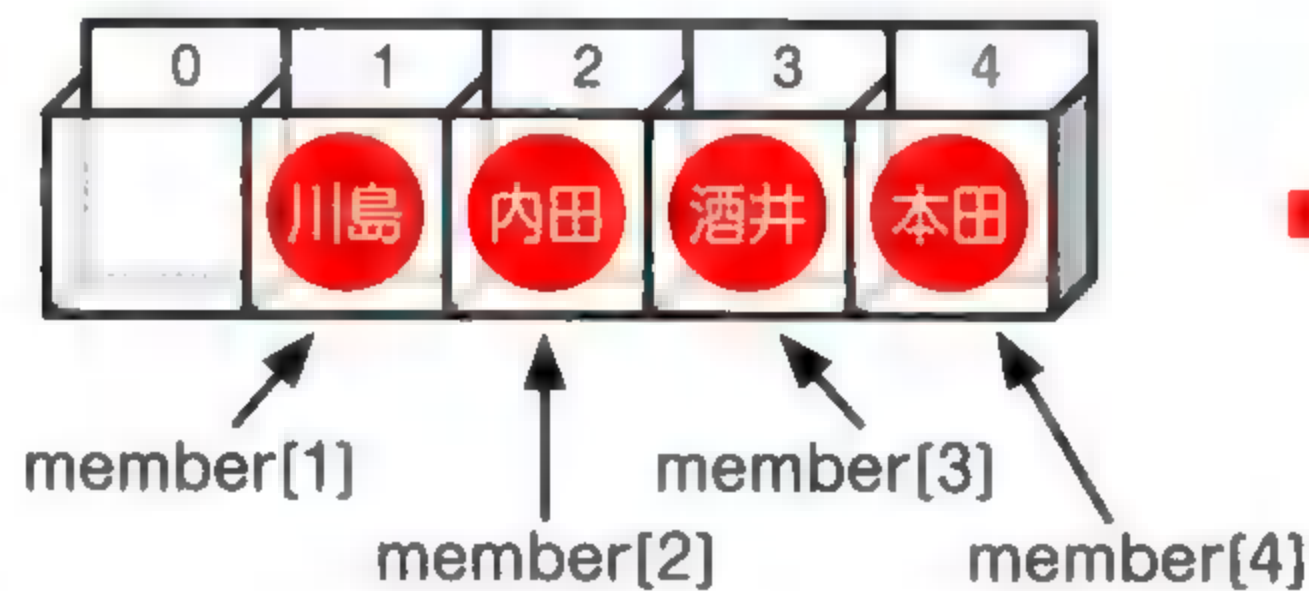
繰り返し文はよく配列と一緒に使われます。配列と for 文を使った典型的な繰り返し処理を見ておきましょう。

●使用する配列とデータ

for 文による繰り返し処理を使って、配列の要素に入っているデータを、1つずつ、順番に出力するプログラムを考えてみたいと思います。

今回使う配列は、サッカー選手4人の名前が、背番号に対応するインデックスの要素に代入されている配列です。この配列のデータを、最終的には次のような文章として出力したいと考えています。

▼ 配列：member



背番号 1 は川島選手
背番号 2 は内田選手
背番号 3 は酒井選手
背番号 4 は本田選手

●配列を宣言してデータを代入

プログラミングに移りましょう。まず変数を宣言して、サッカー選手4人の名前を配列として代入します。要素0は空にしておきます。

```
1 var member = ['', '川島', '内田', '酒井', '本田'];
```





● for 文を使わずに書いてみる

ちょっと回り道になりますが、今回の処理をfor文を使わずに書いてみると、このようになります。

```
1 var member = ['', '川島', '内田', '酒井', '本田'];
2 document.writeln('<p>背番号 1 は ' + member[1] + ' 選手 </p>');
3 document.writeln('<p>背番号 2 は ' + member[2] + ' 選手 </p>');
4 document.writeln('<p>背番号 3 は ' + member[3] + ' 選手 </p>');
5 document.writeln('<p>背番号 4 は ' + member[4] + ' 選手 </p>');
```

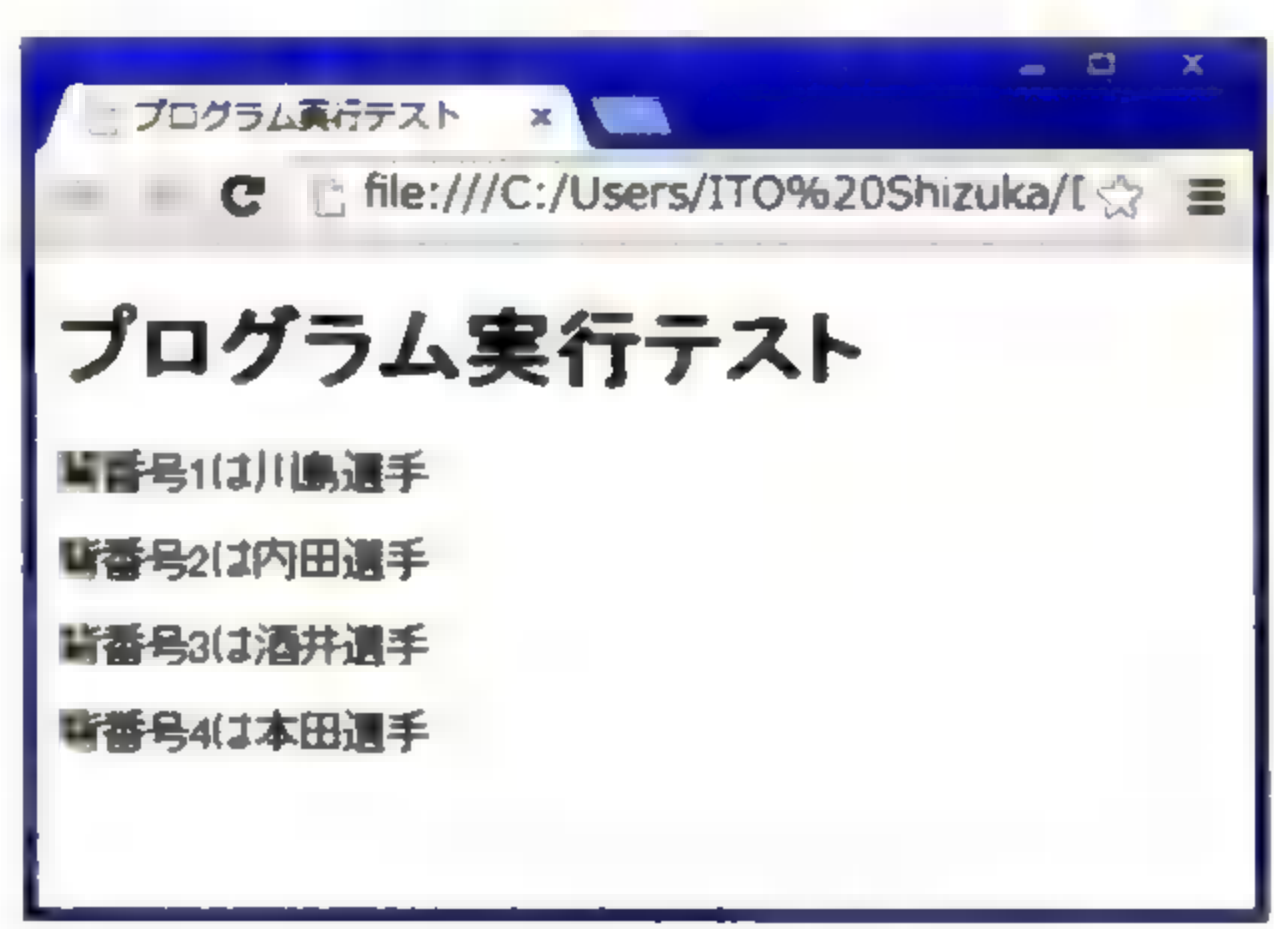
見ていただくと分かるように、背番号とインデックスがきれいに1つずつ増えています。これをfor文のカウンタ変数*i*で置き換えれば、きれいに繰り返し処理ができるはずです。

● 配列のインデックスと for 文のカウンタを連動させる

for文のカウンタ変数*i*を使って、配列のインデックスと要素を表してみます。要素0は出力しないので、*i*の初期値は1にします。

```
1 var member = ['', '川島', '内田', '酒井', '本田'];
2 for (var i = 1; i < 5; i++) {
3     document.writeln('<p>背番号 ' + i + ' は ' + member[i] + ' 選手 </p>');
4 }
```

これでサンプルプログラムは完成です。実行すると次のような画面が表示されます。



背番号と選手名が出力される



インクリメント演算子 とデクリメント演算子

繰り返し文のカウンタを増やしたり減らしたりするときによく使う演算子について説明します。

● インクリメント演算子

インクリメント演算子は、変数の数値を1つ増やす演算子です。インクリメント (increment) は英語で、「増加」という意味です。

演算子	名前	機能
++	インクリメント演算子	変数の数値を 1つ増やす

ソースコードでの使い方はこのようになります。

```
i = 1; // 変数 i に 1 が代入される  
i++; // 変数 i の値が 1 が増えて、2 になる
```

● デクリメント演算子

デクリメント演算子は、変数の数値を1つ減らす演算子です。デクリメント (decrement) は英語で、「減少」という意味です。

演算子	名前	機能
--	デクリメント演算子	変数の数値を 1つ減らす

ソースコードでの使い方はこのようになります。

```
i = 1; // 変数 i に 1 が代入される  
i--; // 変数 i の値が 1 減って、0 になる
```


Day 2 Lesson 3

関数

このレッスンでは、よく使う処理をひとまとめにするための関数のしくみと使い方を学びます。

- 1 関数のしくみ
- 2 戻り値を出力する関数
- 3 引数を受け取る関数
- 4 変数のスコープ
- 5 関数の定義方法の3つのバリエーション



関数のしくみ

関数とは、よく使う処理をひとまとめにして、プログラムの中で何度も使えるようにしたパーツのことです。まずは基本的なしくみと使い方を覚えましょう。

● 関数とは？

関数とは、プログラムの中で何度も使う処理をひとまとめにしたパーツです。自分で関数を作ることを、プログラミング用語で**関数を定義する**といいます。関数を定義するには、function文を使って、このように書きます。

▼ 構文：関数の定義

```
function 関数名() {  
    処理  
}
```

関数名はプログラマが自分で考えて付けます。付け方のルールは、変数名と同じです。関数名のあとに半角丸カッコを付けるのを忘れないようにしてください。

● 関数を定義してみる

ために円の面積を計算する関数を定義してみましょう。まず、ふつうに円の面積を計算する処理は、以下ようになります。半径は7としてあります(わずらわしいので、mやm²などの単位はすべて省略します)。

```
var s = 7 * 7 * 3.14;  
window.alert(s);
```

これを関数にするには、この処理に関数名を付けて、中カッコで囲めば良いのです。



```
1 function circle() {  
2     var s = 7 * 7 * 3.14;  
3     window.alert(s);  
4 }
```

これで関数circle()はできましたが、このままサンプルプログラムを実行しても何も起こりません。関数の処理を実行するには、プログラムの中で「関数を呼び出す」必要があります。

● 関数の呼び出し方

プログラムの中で関数を使うことを、プログラミング用語で**関数を呼び出す**といいます。関数を呼び出すには、関数名と半角丸カッコを書くだけでOKです。

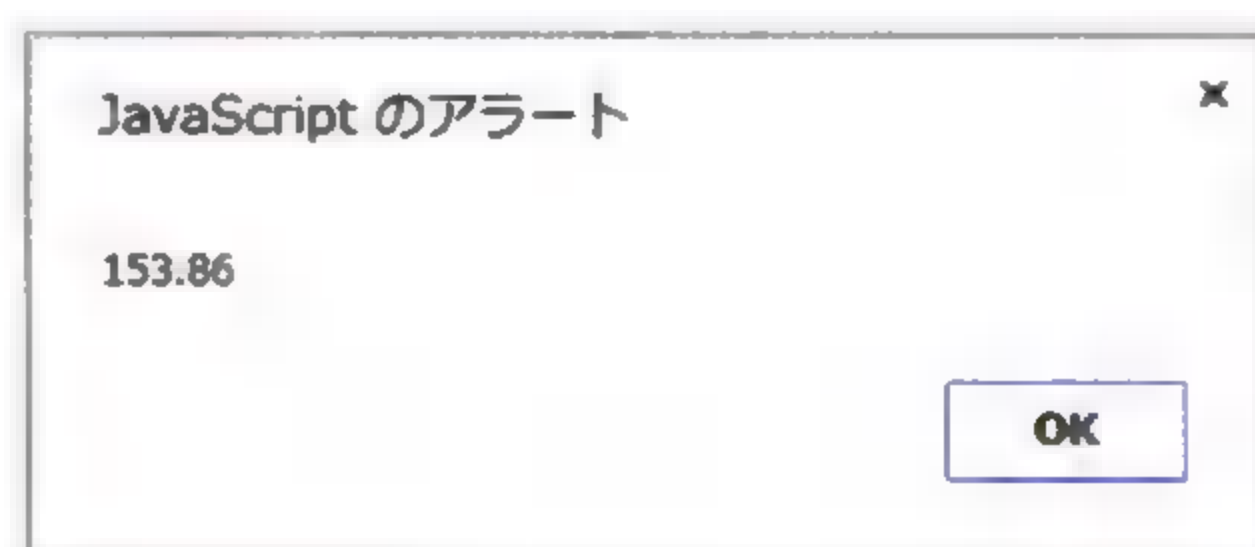
▼ 構文：関数の呼び出し方

関数名 ()

● 定義した関数を呼び出すサンプルプログラム

先ほど作ったサンプルプログラムに、定義した関数を呼び出す1行を書き加えて(5行目)、実行してみましょう。ダイアログボックスに面積の数値が表示できれば成功です。

```
1 function circle() {  
2     var s = 7 * 7 * 3.14;  
3     window.alert(s);  
4 }  
5 circle();
```



円の面積が表示される



戻り値を出力する関数

関数の大事な機能である戻り値のしくみと使い方を学びます。

● 関数を改良してみる

前のセクションで作った関数 `circle()` では、円の面積を計算して、ダイアログボックスに出力しました。今回はこれを少し改良して、関数の処理を「円の面積を計算する」ということに特化させてみましょう。つまり、このような形になります。

```
1 function circle() {  
2     var s = 7 * 7 * 3.14;  
3 }  
4 circle();
```

しかしこのままでは、関数 `circle()` を呼び出しても円の面積は表示されません。関数を呼び出すときに、面積を表示する処理を書き加える必要があります。面積を警告ダイアログボックスに表示したいので、4行目をこのように変えてみましょう。

```
1 function circle() {  
2     var s = 7 * 7 * 3.14;  
3 }  
4 window.alert(circle());
```

これで実行したところ、ダイアログボックスには「undefined」と表示されるだけで、円の面積は表示されません。

ではどうすればいいか？ 関数の処理結果をプログラムの他の場所で使うには、関数から**戻り値 (もどりち)** が出力されるように定義する必要があります。

● 戻り値とは？

戻り値とは、「関数から出力されるデータ」のことです。戻り値を指定してある関数を呼び出すと、内部の処理が実行され、その処理結果が戻り値（データ）として出力されます。

戻り値というとなんだか特別な感じがしますが、実体は普通のデータなので、他のデータと同じように使うことができます。

▼ 戻り値を出力する関数のしくみ



今回のサンプルプログラムでは、変数sの値を戻り値として出力したいと考えています。戻り値を指定するには**return文**を使います。return文は、戻り値を出力して、関数の処理を終える働きをします。書き方ですが、「return」のあとに、戻り値として出力したい変数名や式を記入するだけです。これで変数の値や式の結果が戻り値として出力されます。

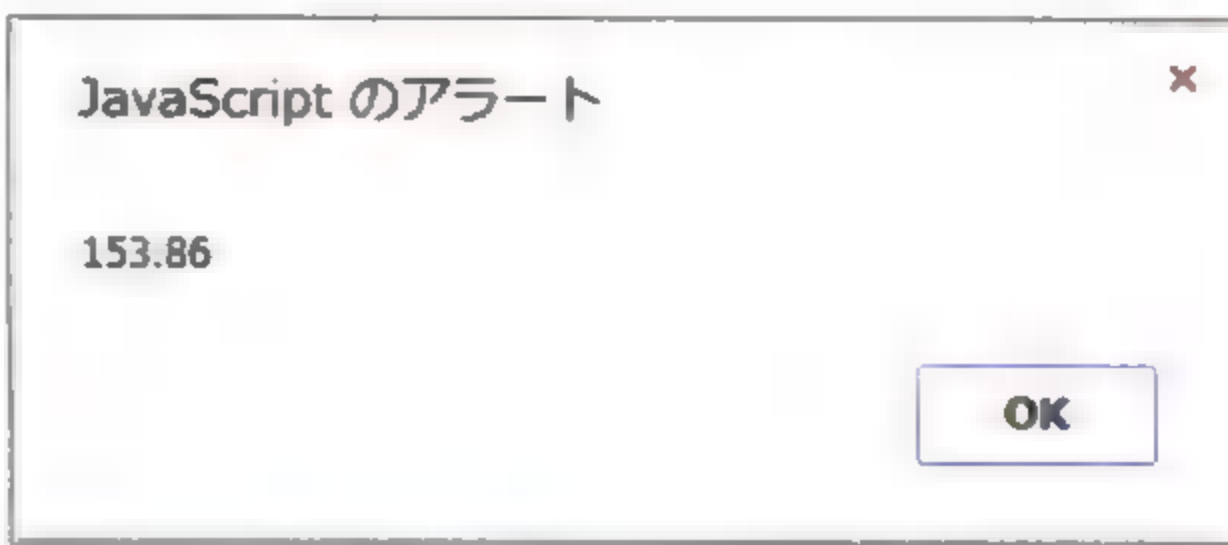
▼ 構文：戻り値を出力する関数の定義

```
function 関数名() {  
    処理  
    return 変数名や式;  
}
```

今回のプログラムでは、変数sの値を戻り値として出力するので、3行目に「return s;」という1行を追加します。

```
1 function circle() {  
2     var s = 7 * 7 * 3.14;  
3     return s;  
4 }  
5 window.alert(circle());
```

このプログラムを実行すると、警告ダイアログボックスに円の面積が表示されます。



円の面積が表示された

MEMO

if文やfor文と違って、関数の中カッコ{}の中はブロックではないので、文が1つだけの場合でも{}を省略してはいけません。

MEMO

戻り値は、**返り値(かえりち)**と呼ばれることもあります。
1つの関数につき、戻り値は1つしか指定できません。

returnのあとには、変数を使わずに、式を直接書いてもかまいません。たとえば、今回のサンプルプログラムは、このように書いても動作は同じです。

```
1  function circle() {  
2      return 7 * 7 * 3.14;  
3  }  
4  window.alert(circle());
```


Lesson 3



引数を受け取る関数

関数の大事な機能である引数（ひきすう）のしくみと使い方を学びます。

Lesson 3



引数を受け取る関数

● 関数内の変数にデータを代入する

今のところ、関数circle()は半径7の円の面積を計算して出力する関数です。しかし、できれば、もっといろいろな半径値の円の面積を計算できる、汎用的な関数にしたいところです。

たとえば、半径の数値を変数に置き換えて、「関数の中の変数に値を自由に代入できるような関数」にできないでしょうか？ 半径7を変数rに置き換えてみましょう。

```
1 function circle() {
2   var s = r * r * 3.14;
3   return s;
4 }
```

このようにした場合、変数rの値をどこかで代入できるようにしなくてははいけません。関数内部の変数に値を代入するには、**引数（ひきすう）**というしくみを使います。

● 引数とは？

引数とは、「関数に入力するデータ」のことです。呼び出された関数が、引数を受け取れるようにするには、このように定義します。

```
1 function circle(r) {
2   var s = r * r * 3.14;
3   return s;
4 }
```



関数名の半角丸カッコの中に、引数を代入したい変数名（今回のサンプルではr）を記入します。この変数を**仮引数（かりひきすう）**といいます。

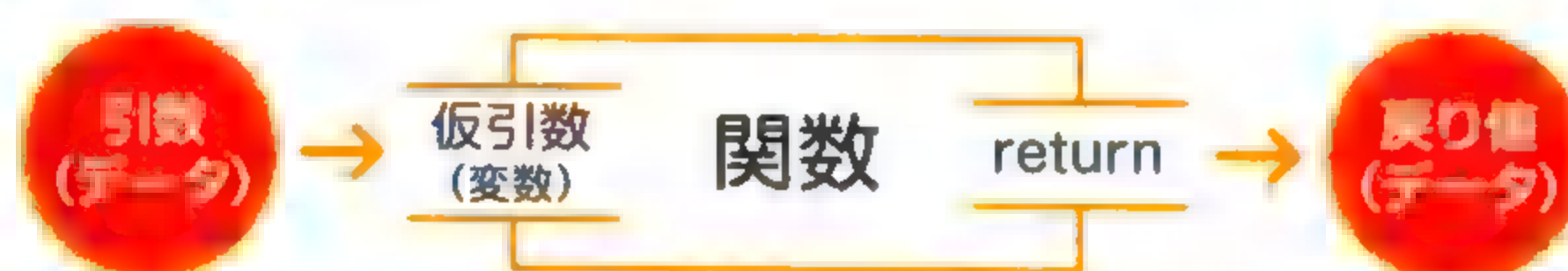
▼ 構文：引数を受け取る関数の定義

```
function 関数名(仮引数) {  
    処理  
    return 変数名や式;  
}
```

MEMO

仮引数は、変数ですが、varで宣言する必要はありません。

▼ 引数を受け取る関数のしくみ



MEMO

仮引数は「パラメータ (parameter)」とも呼ばれます。また、引数のことを「実引数 (じつひきすう)」と呼ぶこともあります。

● 関数を呼び出して引数を渡すには？

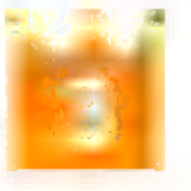
呼び出す関数に引数を渡すには、関数の半角丸カッコの中に、引数として渡したいデータを記入すればOKです。

▼ 構文：引数を渡して関数を呼び出す方法

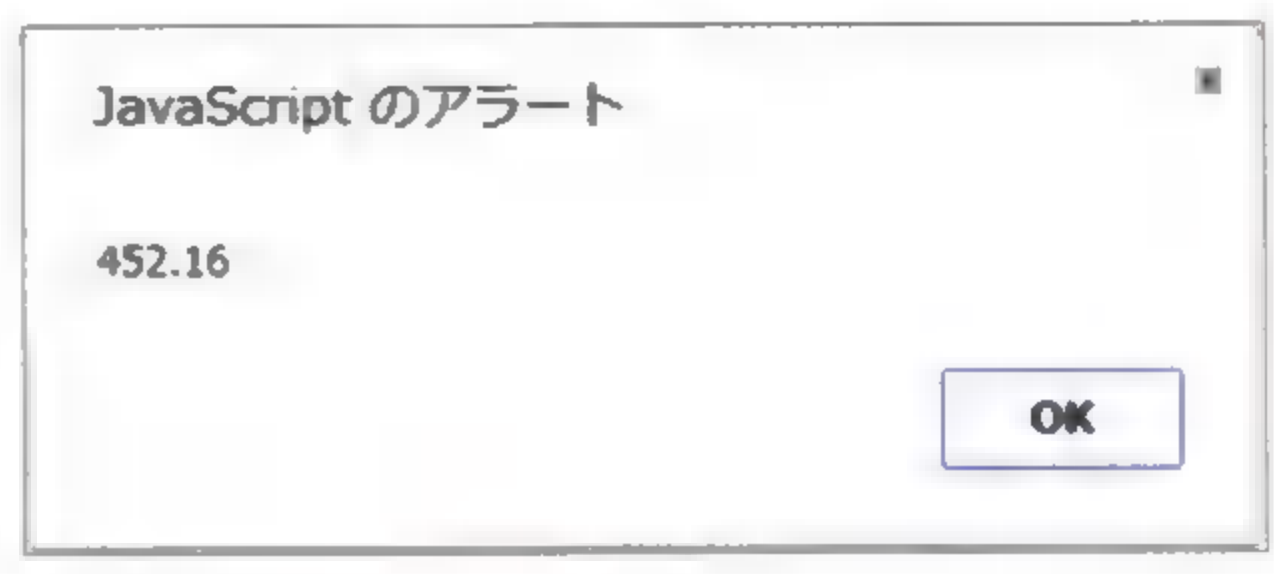
```
関数名(引数)
```

たとえば、半径12の円の面積を計算して出力してほしいときは、「circle(12)」と書きます(5行目)。

```
1 function circle(r) {  
2     var s = r * r * 3.14;  
3     return s;  
4 }  
5 window.alert(circle(12));
```

このプログラムを実行すると、アラートダイアログボックスに円の面積が表示されます。



半径 12 の円の面積が表示された

● 仮引数は複数指定できる

仮引数はいくつでも指定できます。関数を定義するときに、仮引数を複数指定する場合は、それぞれの仮引数を半角カンマで区切ります。

▼ 構文：複数の引数を使う関数の定義

```
function 関数名(仮引数1, 仮引数2) {
    処理
    return 変数名や式;
}
```

呼び出した関数に複数の引数を渡すには、関数の半角丸カッコの中に、引数として渡したいデータを半角カンマで区切って記入します。

▼ 構文：複数の引数を渡して関数を呼び出す方法

```
関数名(引数1, 引数2)
```

たとえば、四角形の面積を計算する関数squareを作ってみましょう。仮引数には、変数w(底辺の長さ)と変数h(高さ)の2つを指定します(1行目)。関数を呼び出すときに、底辺の長さと高さを引数として渡せば(5行目)、四角形の面積がダイアログボックスに表示されます。

```
1 function square(w, h) {
2     var s = w * h;
3     return s;
4 }
5 window.alert(square(7, 5));
```



変数のスコープ

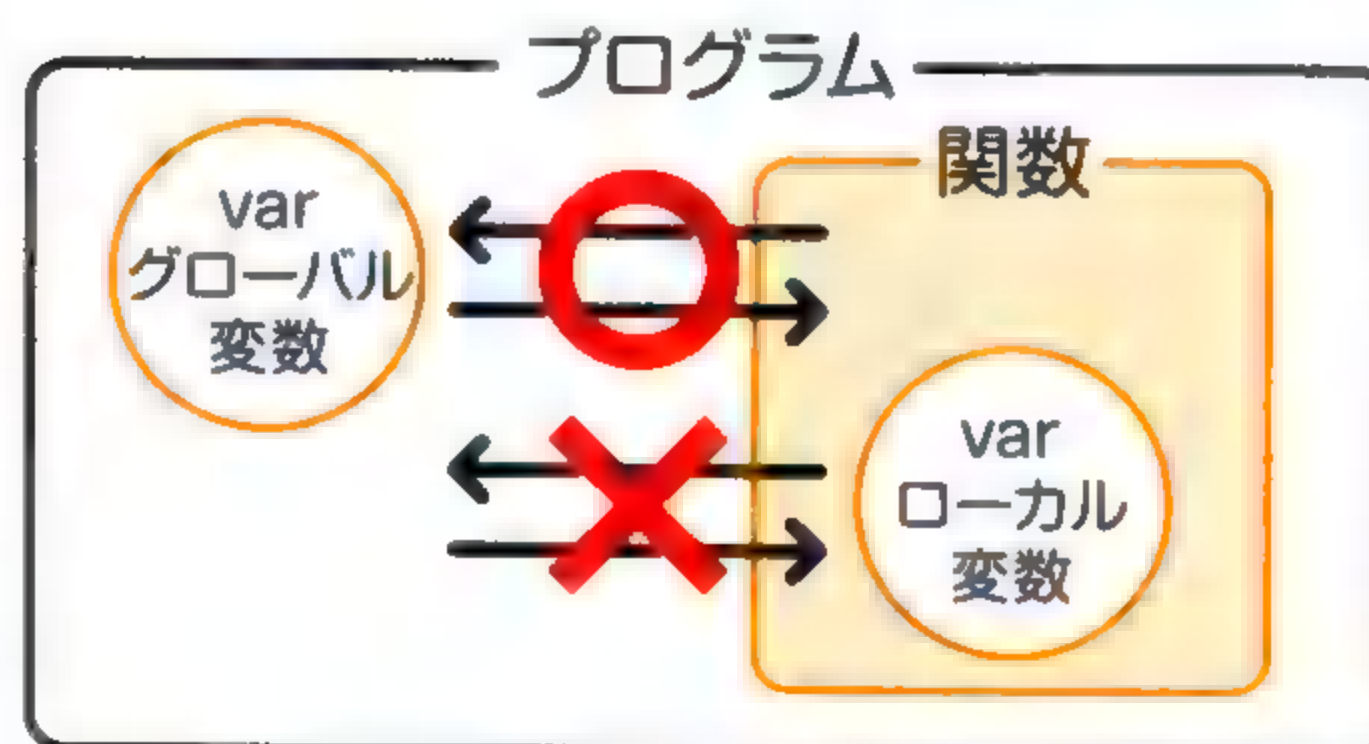
変数の有効範囲のことをスコープといいます。関数内部の変数にかかわる事柄ですので、ここで説明しておきます。

●グローバル変数とローカル変数

変数のスコープ（有効範囲）とは、**その変数が存在していて、使用できる範囲**のことです。プログラムで宣言した変数のスコープは、そのプログラム全体です。プログラムの中に関数がある場合は、関数の内部でもその変数に値を代入したり、参照したりすることができます。プログラム全体をスコープとする変数のことを、**グローバル（大域）変数**といいます。

一方、関数の内部でvarで定義した変数のスコープは、その関数の内部だけになります。関数の外では、その変数に値を代入したり、参照したりすることはできません。このような関数の内部だけをスコープとする変数のことを、**ローカル（局所）変数**といいます。

関数の仮引数は、varで定義していませんが、自動的にローカル変数となります。



ローカル変数は
関数外部からは使えない

COLUMN

関数の内部でグローバル変数を宣言するには？

関数の内部の変数をグローバル変数として宣言することもできます。グローバル変数として宣言するには、varを付けずに宣言すればOKです（要するに、宣言せずに使う）。こうすれば関数の外から使うことができます。



関数の定義方法の 3つのバリエーション

JavaScriptには、すでに紹介した方法以外にも関数を定義する方法があります。最後にそれらをまとめて説明しておきます。初級の段階では使わないかもしれませんが、とりあえず知っておきましょう。



● 宣言して定義する

まずおさらいです。このレッスンで最初に紹介したのは、function文と関数名と使う方法でした。この方法で定義された関数を**宣言型の関数**といいます。

▼ 構文：宣言型の関数定義

```
function 関数名() {  
    処理  
}
```

● 変数に代入して定義する

2つ目は変数を使う方法です。関数は変数に代入する方法でも定義することができます。この右辺の書き方を**関数リテラル**といいます。

▼ 構文：変数を使った関数定義

```
var 変数名 = function() {  
    処理  
}
```

この関数は、functionのあとに関数名を書かないので、**無名関数**とも呼ばれます。配列やオブジェクトなどと同じで、代入した変数名が関数名になります。関数を呼び出す方法や、引数を渡す方法は、宣言型の関数と同じです。たとえば、2-3-3の関数の定義は、このようにも書けます。

```
1 var circle = function(r) {  
2     var s = r * r * 3.14;
```



```
3     return s;  
4 }
```

2つ目の方法の応用ですが、変数に代入して定義できるということは、オブジェクトのプロパティに代入しても定義できます。変数shapeにオブジェクトを代入し、circleプロパティに関数を代入するには、このように書きます。

```
1  var shape = {  
2    circle: function(r) {  
3      var s = r * r * 3.14;  
4      return s;  
5    }  
6  };  
7  window.alert(shape.circle(6));
```

プロパティに代入した関数は、**メソッド**と呼ばれます(1-6-1で軽く触れました)。すでにサンプルプログラムの中には書いてありますが、メソッドを呼び出すには、オブジェクト名とピリオド(.)を使って、プロパティを参照するのと同じ書き方をします。

▼メソッドの呼び出し方

オブジェクト名.メソッド名()

●コンストラクタを使って定義する

3つ目はコンストラクタを使う方法です。2つ目と同じように変数に代入して定義しますが、右辺の書き方がちょっと違います。Functionの先頭文字は大文字です。

▼構文：コンストラクタを使った関数の定義方法

var 変数名 = new Function('仮引数', '処理');

このように定義された関数も**無名関数**です。コンストラクタとは何なのか、これはいったいどういうしくみなのか、くわしくは2-4で説明しますので、とりあえず存在だけ覚えておいてください。



Day 2 Lesson 4

組み込みオブジェクト

このレッスンでは、JavaScriptにあらかじめ用意されている組み込みオブジェクトについて説明します。ちょっとプログラミング的に高度な内容なので、難しいなと思った方は、飛ばし読みしていただいてもかまいません。

- 1 組み込みオブジェクトのしくみ
- 2 Date オブジェクト
- 3 Math オブジェクト
- 4 RegExp オブジェクト
- 5 Array オブジェクト
- 6 String オブジェクト
- 7 Number オブジェクト
- 8 Function オブジェクト
- 9 その他の組み込みオブジェクト



組み込みオブジェクトのしくみ

SECTION

1

JavaScriptには、オブジェクトのテンプレートである組み込みオブジェクトがあらかじめ用意されています。

●組み込みオブジェクトとは？

JavaScriptには、あらかじめよく使うオブジェクトのテンプレート（ひな形）が用意されています。これらオブジェクトのテンプレートは、**組み込みオブジェクト**と呼ばれます。

MEMO

組み込みオブジェクトは、他にも「ビルトインオブジェクト」「ネイティブオブジェクト」「固有のオブジェクト」「基本のオブジェクト」などと呼ばれたりもします。

●組み込みオブジェクトの種類

組み込みオブジェクトには、以下のような種類があります。このあと1つずつ説明しますので、とりあえずどんなものがあるのかだけ、ざっと目を通しておいってください。

オブジェクト名	機能
Date オブジェクト	日付や日時 を扱うオブジェクト
Math オブジェクト	数学 関連の機能をまとめているオブジェクト
RegExp オブジェクト	正規表現 を使うためのオブジェクト
String オブジェクト	文字列 に対応するオブジェクト
Number オブジェクト	数値 に対応するオブジェクト
Boolean オブジェクト	論理値 に対応するオブジェクト
Object オブジェクト	すべての オブジェクトの基本 となるオブジェクト
Array オブジェクト	配列 に対応するオブジェクト
Function オブジェクト	関数 に対応するオブジェクト
Error オブジェクト	エラー に関するオブジェクト



● 組み込みオブジェクトを使って新しいオブジェクトを作るには？

組み込みオブジェクトを使って、新しいオブジェクトを作りたいときは、**new** 演算子を使って、変数を宣言します。

▼ 構文：オブジェクトの作り方

```
var 変数名 = new 組み込みオブジェクトの種類();
```

↑
コンストラクタ

MEMO

「new 組み込みオブジェクトの種類()」の部分を、プログラミング用語で、**コンストラクタ**といいます。

たとえば、新しいDateオブジェクトを生成して、変数todayに代入するには、このように書きます。

```
var today = new Date();
```

これで、変数todayの中には、Dateオブジェクトのテンプレートによって作られた、新しいオブジェクト（インスタンス）が代入されています（Dateオブジェクトについては、次のセクションで説明します）。

MEMO

組み込みオブジェクトをもとに作られた新しいオブジェクトのことを、プログラミング用語で、**インスタンス**といいます。

● 組み込みオブジェクトの何が便利なのか？

扱いたいデータの種別に応じた適切な組み込みオブジェクトを使えば、自分でゼロからオブジェクトを作るのに比べて、プロパティやメソッドを準備する手間が大幅に軽減されます。



Lesson 4

SECTION

2

Date オブジェクト

Date オブジェクトは、日付や時刻に関するプロパティ（おもにメソッド）があらかじめ備わっている組み込みオブジェクトです。

● Date オブジェクトのインスタンスを作成する

Date オブジェクトのコンストラクタを使って、新たなオブジェクト（インスタンス）を作るには、このように書きます。

▼ 構文：Date オブジェクトのインスタンスの作成

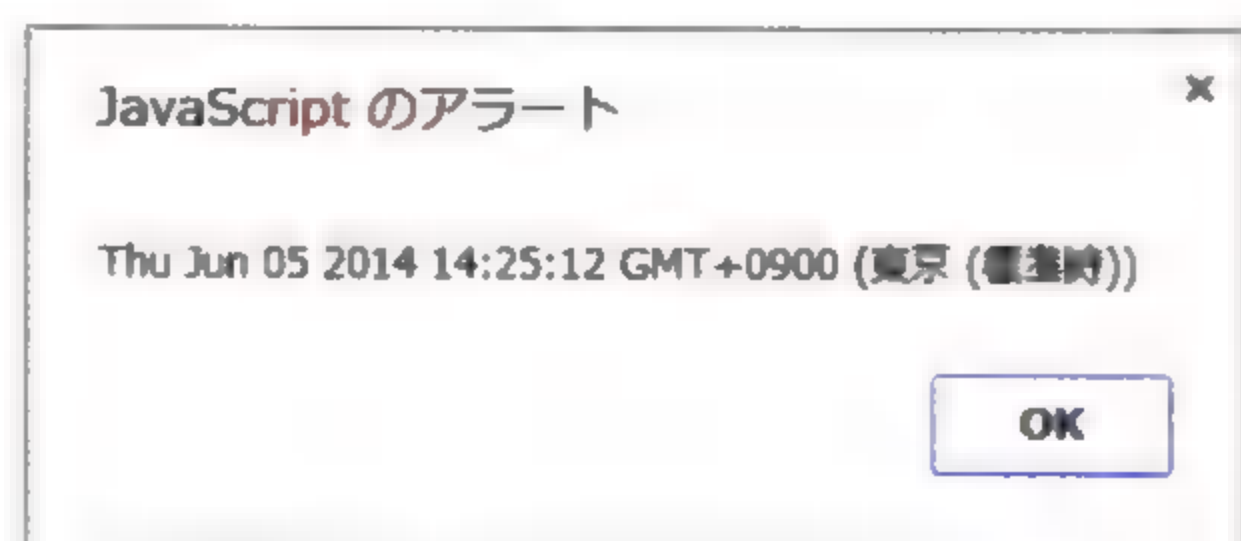
```
var 変数名 = new Date();
```

● Date オブジェクトを使ったサンプル

実際にDateオブジェクトのインスタンスを作ってみましょう。以下のソースコードを入力して、実行してみてください。

```
var today = new Date();  
window.alert(today);
```

実行すると、実行した日時が書かれたアラートダイアログボックスが表示されます。



実行時の日時が表示された

● サンプルプログラムの解説

today という変数には Date オブジェクトのインスタンスが作られていて、作成された日時に関するデータがすでに代入されています。今回は「2014年6月5日14時25分12秒」に、この命令を実行したので、代入されているデータが加工されて「Thu Jun 05 2014 14:25:12 GMT +0900 (東京(標準時))」と表示されたというわけです。

● Date オブジェクトのおもなメソッド

Date オブジェクトには、次のようなメソッドがあります。

▼ Date オブジェクトのおもなメソッド

メソッド名	機能
getTime()	日時を 1970 年 1 月 1 日 0 時以降のミリ秒として返す
getFullYear()	年の値 (4 桁) ローカル時間で返す
getMonth()	月の値 (0 ~ 11) をローカル時間で返す
getDate()	日付の値 (1 ~ 31) をローカル時間で返す
getDay()	曜日の値 (0 ~ 6) をローカル時間で返す
getHours()	時の値 (0 ~ 23) をローカル時間で返す
getMinutes()	分の値 (0 ~ 59) をローカル時間で返す
getSeconds()	秒の値 (0 ~ 59) をローカル時間で返す
getMilliseconds()	ミリ秒の値 (0 ~ 999) をローカル時間で返す
toDateStrin()	ローカル時間の日付を文字列として返す
toString()	ローカル時間の日時を文字列で返す
toTimeString()	ローカル時間の時刻を文字列で返す
toLocaleDateString()	ローカル時間の日付をローカル表現した文字列で返す
toLocaleString()	ローカル時間の日時をローカル表現した文字列で返す
toLocaleTimeString()	ローカル時間の時をローカル表現した文字列で返す
setDate()	日付の値 (1 ~ 31) をローカル時間で設定する
setFullYear()	年の値 (4 桁) をローカル時間で設定する
setHours()	時の値 (0 ~ 23) をローカル時間で設定する
setMilliseconds()	ミリ秒の値 (0 ~ 999) をローカル時間で設定する
setMinutes()	分の値 (0 ~ 59) をローカル時間で設定する
setMonth()	月の値 (0 ~ 11) をローカル時間で設定する
setSeconds()	秒の値 (0 ~ 59) をローカル時間で設定する



MEMO

ローカル時間とは、世界各地の標準時のことです。標準時は協定世界時（UTCまたはGMT）を基準に決められています。たとえば日本標準時（JST）は協定世界時より9時間進んでいるので「UTC（GMT）+0900」と表記されます。

Dateオブジェクトのインスタンスに入っている日時に関するデータを参照したり、データを上書きしたりするには、用意されているメソッドを使います。

たとえば、これらのメソッドを使って、今日は「〇月〇日です」という文字列を表示したいときには、このようなソースコードを書きます。

▼ 現在の月日を表示するサンプルプログラム

```
1 var today = new Date();
2 var m = today.getMonth() + 1;
3 var d = today.getDate();
4 window.alert('今日は' + m + '月' + d + '日です');
```

月と日付のデータは、それぞれいったん変数mと変数dに代入してから（2、3行目）、警告ダイアログボックスに表示しています（4行目）。

月のデータは1月が「0」なので、あらかじめ1を足して（2行目）、私たちがふだん使っている月数が正しく表示されるようにしてあります。

● 特定の日時データを持つインスタンスを作るには？

Dateオブジェクトは、現在の日時を自動的に取得したインスタンスだけでなく、ある特定の日時データを持つインスタンスを作することもできます。

▼ 構文：特定の日時データを持つインスタンスの作成

var 変数名 = new Date(年, 月, 日, 時, 分, 秒, ミリ秒);

たとえば、「2014年7月25日午後2時30分」というデータが代入されたインスタンスを作成するには、このように書きます。

```
1 var birthday = new Date(2014, 6, 25, 14, 30);
2 window.alert(birthday);
```




Lesson 4



Math オブジェクト

Math オブジェクトは、数学関連の機能をまとめている組み込みオブジェクトです。プログラムで数学的な操作をするときに使います。

● Math オブジェクトはインスタンスを作らない

Mathオブジェクトの使い方は、他の組み込みオブジェクトとは違います。コンストラクタを使ってインスタンスを作る必要はありません。インスタンスを作らずに、すぐに備わっているプロパティやメソッドを使えます。

● Math オブジェクトのプロパティとメソッド

Mathオブジェクトには、次のようなプロパティとメソッドがあります。

▼ Math オブジェクトのプロパティ

プロパティ名	説明
E	数学定数 e (ネイピア数、オイラー数)。自然対数の底
LN2	2 の自然対数
LN10	10 の自然対数
LOG2E	2 を底とした e の対数
LOG10E	10 を底とした e の対数
PI	円周率、パイ (π)
SQRT1_2	1/2 の平方根
SQRT2	2 の平方根



▼ Math オブジェクトのメソッド

メソッド名	機能
round(x)	x の小数第 1 位を四捨五入した値を返す
ceil(x)	x の小数点以下を切り上げる
floor(x)	x の小数点以下を切り捨てる
abs(x)	x の絶対値を返す
pow(x, y)	x を y で累乗した値を返す
sqrt(x)	x の平方根を返す
log(x)	x の自然対数を返す
exp(x)	数学定数 e を x で累乗した数を返す
max(x, y)	引数の中で最大の値を返す
min(x, y)	引数の中で最小の値を返す
random()	0 ~ 1 の範囲の擬似乱数を返す
sin(x)	x のサインを返す
cos(x)	x のコサインを返す
tan(x)	x のタンジェントを返す
asin(x)	x のアークサインを返す
acos(x)	x のアークコサインを返す
atan(x)	x のアークタンジェントを返す
atan2(y, x)	x 軸から、指定された y 座標と x 座標で表された点までのアークタンジェントを返す

Math オブジェクトのメソッドを使用したサンプルコードは以下のとおりです。

```
//3.7 を小数第 1 位で四捨五入
document.writeln(Math.round(3.7)); // 戻り値は 4
//3.7 の小数点以下を切り捨て
document.writeln(Math.floor(3.7)); // 戻り値は 3
//2 の 4 乗
document.writeln(Math.pow(2, 4)); // 戻り値は 16
// 最大値
document.writeln(Math.max(-6, 19)); // 戻り値は 19
// 最小値
document.writeln(Math.min(-6, 19)); // 戻り値は -6
// 乱数
document.writeln(Math.random()); // 戻り値例 0.05458003003150225
```


Lesson 4

SECTION

4

RegExp オブジェクト

RegExp オブジェクトは、正規表現を使うための組み込みオブジェクトです。ここでは、正規表現を使った検索の基本を学びます。

● 正規表現とは？

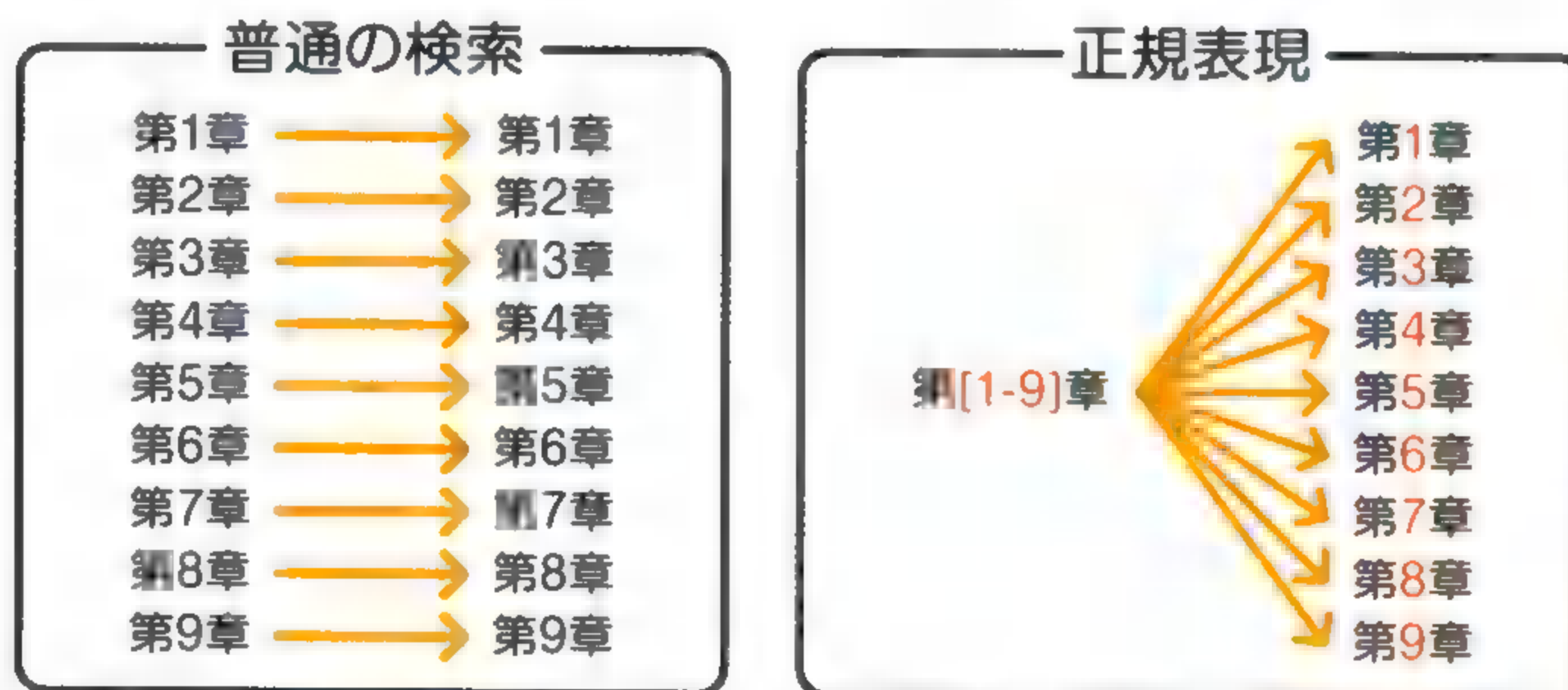
正規表現は、文字列の検索や置換を何倍も楽にする方法です。なぜ楽になるのかというと、ある「特殊な書き方」を使うことで、通常なら何回にも分けて検索しなければならないところを、一度でズバッと検索できるからです。この「特殊な書き方」のことを正規表現といいます。

MEMO

オブジェクト名の RegExp とは、**Regular Expression**（正規表現）の略です。

たとえば、ある文章の中から「第1章、第2章、…、第9章」という文字列を検索したいとします。普通の検索では、まず「第1章」を検索し、次に「第2章」を検索し、それから順番に「第9章」まで、9回検索しなければなりません。

しかし、正規表現の書き方を使って「第[1-9]章」という文字列で検索すれば、1回の検索で「第1章」から「第9章」までを一度に検索できるのです。



この「第[1-9]章」という文字列の中の[1-9]は、「1から9までの数字」という意味を表している正規表現です。このような特殊な書き方を正規表現の**パターン**といいます。

正規表現には、検索や置換に便利なパターンがいくつも用意されています。

● 正規表現のパターン一覧

JavaScriptで使えるおもな正規表現のパターンを以下に挙げておきます。

▼ おもな正規表現パターン

表記	説明
^	行の先頭
\$	行の末尾
x*	直前の文字 x の 0 回以上の繰り返し
x+	直前の文字 x の 1 回以上の繰り返し
x?	直前の文字 x の 0 回か 1 回の出現
x{n}	直前の文字 x の n 回の繰り返し
x{n,m}	直前の文字 x の n ~ m 回の繰り返し
.	何か 1 文字
x y z	x または y または z
[xyz]	xyz のうちのいずれかの 1 文字
[^xyz]	xyz 以外の 1 文字
[a-z]	a から z のあいだの 1 文字 (文字コード順)
¥b	単語の区切り
¥B	単語の区切り以外の文字
¥d	数字。[0-9] と同じ
¥D	数字以外の文字。[^0-9] と同じ
¥n	改行文字
¥s	1 つのホワイトスペース (スペース、タブ、改行文字)
¥S	ホワイトスペース以外の 1 文字
¥t	タブ
¥w	英数字。[A-Za-z0-9_] と同じ
¥W	英数字以外の文字。[^A-Za-z0-9_] と同じ

MEMO

正規表現についてもっと知りたい方は、別途、正規表現の参考書やWebサイトなどで学んでください。

● RegExpオブジェクトのインスタンスを作成する

RegExpオブジェクトのインスタンスには、検索文字列の正規表現パターンを代入できます。

RegExpオブジェクトのインスタンスを作る方法は、2種類あります。まずはコンストラクタを使う方法です。

▼ 構文：コンストラクタでインスタンスを作成

```
var 変数名 = new RegExp('正規表現', 'フラグ');
```

フラグ	指定した場合	指定しない場合
g	マッチする箇所すべてを検索・置換する	1カ所マッチしたら終了する
i	大文字と小文字を区別しない	大文字と小文字を区別する
m	行頭、行末を無視する	行頭、行末を認識する

フラグとしてgを指定すると、検索対象となっている文字列の中に複数マッチする箇所があれば、すべてにマッチします。指定しない場合は、最初の1カ所にマッチしたら、そこで検索は終了します。

iを指定すると、正規表現にアルファベットが含まれている場合は、大文字にも小文字にもマッチします。たとえば「[a-z]」という正規表現であれば、小文字のa～zと大文字のA～Zにマッチします。

mを指定すると、複数行にまたがった箇所にもマッチします。

MEMO

フラグは複数を同時に指定することもできます。その場合は、「gi」や「gim」などと書きます。どの順番に書いてもかまいません。

インスタンスを作るもう1つの方法は、スラッシュ (/) を使って書く方法です。

▼ 構文：正規表現リテラルを使ったインスタンス作成

```
var 変数名 = /正規表現/フラグ;
```

このような書き方を**正規表現リテラル**といいます。普通はこちらの方法を使うことが多いようですので、この書き方を覚えてください。フラグの指定のしかたは、コンストラクタを使う場合と同じです。



● RegExp オブジェクトのメソッド

RegExpオブジェクトには、次の2つのメソッドがあります。

▼ メソッド

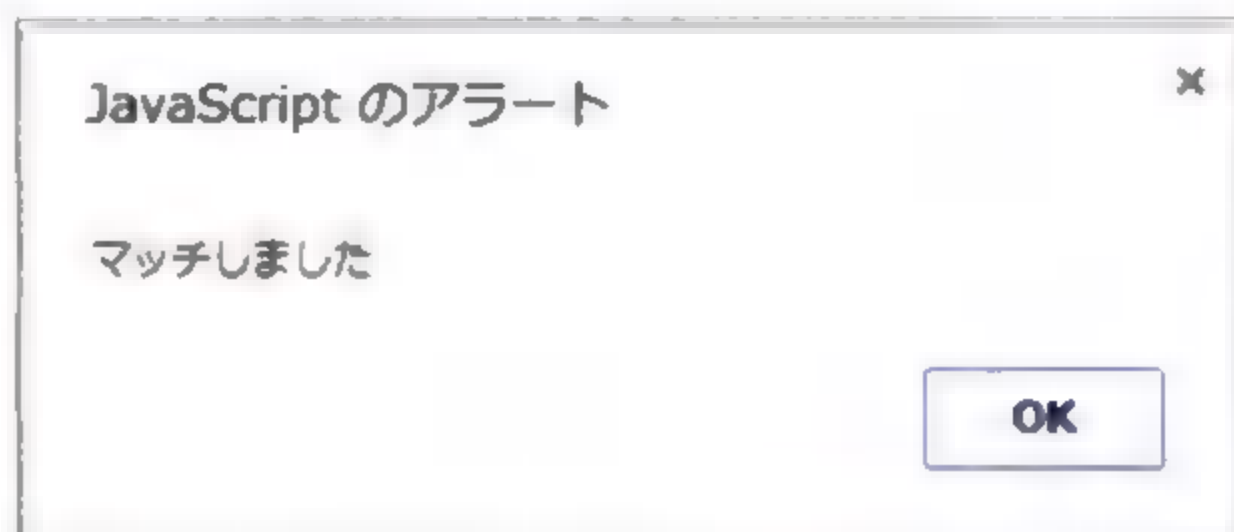
メソッド名	機能
test()	引数の文字列パターンと一致するものがあれば true、なければ false を返す
exec()	引数の文字列パターンと一致するものがあれば、配列として返す

● test() メソッドを使ったサンプルプログラム

test() メソッドを使って、正規表現パターンで検索してみましょう。以下のソースコードを入力して、実行してみてください。

```
1 var r = /[1-9] 章 /;  
2 var book = 'この本は第 1 章から第 9 章まであります。';  
3 if (r.test(book)){  
4     window.alert(' マッチしました ');  
5 }
```

実行すると、「マッチしました」と書かれた警告ダイアログボックスが表示されます。



「マッチしました」と表示される

1行目で、変数rに正規表現パターンを代入しています。このrはRegExpオブジェクトのインスタンスです。

2行目は、検索される対象となる文字列を、変数bookに代入しています。

3～5行目はif文です。条件は「r.test(book)」という式で、変数r (RegExpオブジェクト) のtestメソッドを呼び出して、引数に変数bookを指定しています。つまり、bookの文字列の中に「第[1-9]章」という正規表現パターンと一致するものがあればtrue、なければfalseを返す、というわけです。

今回はtrueですので、警告ダイアログボックスを表示して終了します(4行目)。



Array オブジェクト

Array オブジェクトは、配列を扱うのにいろいろ便利なメソッドやプロパティが用意されている組み込みオブジェクトです。



● Array オブジェクトのインスタンスは配列

Array オブジェクトのインスタンスというのは、すでに学んだ配列と同じものです。コンストラクタを使って、要素を持たない空のインスタンス（配列）を作るには、このように書きます。

▼ 構文：空のインスタンスを作る方法

```
var 変数名 = new Array();
```

要素に代入するデータを指定してインスタンスを作る場合は、引数にカンマで区切ってデータを記入します。こうすれば、データが先頭要素から順番に代入された状態で配列が作成されます。

▼ 構文：要素に代入するデータを指定する方法

```
var 変数名 = new Array( データ1, データ2, データ3 );
```

とりあえず要素の数を指定してインスタンスを作る場合は、引数に要素の数を指定します。

▼ 構文：要素に代入するデータを指定する方法

```
var 変数名 = new Array( 要素の数 );
```



● コンストラクタを使わずに作るには？

しかし、配列を作るのであれば、すでに学んだ大カッコとカンマ(,)を使う方法で十分です。この書き方を**配列リテラル**といいます。

```
var 変数名 = [データ1, データ2, データ3];
```

JavaScriptは、配列リテラルで作った配列でも、Arrayオブジェクトのメソッドやプロパティが使えるような親切なしくみになっています。

● Array オブジェクトのプロパティとメソッド

Arrayオブジェクトのインスタンス（と配列リテラルを代入した変数）で使える、おもなプロパティとメソッドです。すでに1-5で説明したものもありますが、まとめて挙げておきます。

▼ おもなプロパティ

プロパティ名	機能
length	配列の要素の数を返す

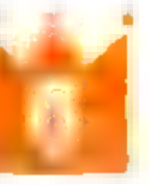
▼ おもなメソッド

メソッド名	機能
sort()	要素の順番を並べ替える
reverse()	要素の並び順を逆にする（元の配列が変更される）
shift()	先頭の要素を削除し、削除した要素を返す
pop()	末尾の要素を削除し、削除した要素を返す
splice()	要素を削除して、削除した要素を返す
slice()	要素を、範囲を指定して取り出す（元の配列は変更されない）
unshift()	先頭に新しい要素を挿入し、追加後の配列の長さを返す
push()	末尾に要素を追加し、追加後の配列の長さを返す
concat()	末尾に、複数の要素や別の配列を結合した新しい配列を返す
toString()	全要素を、カンマで区切って結合した文字列を返す
join()	全要素を、カンマなどで区切って結合した文字列を返す

通常、配列を作るときには、書き方が簡潔で内容が明解な配列リテラルを使ってください。



String オブジェクト



String オブジェクトは、文字列を扱うのにいろいろ便利なメソッドやプロパティが用意されている組み込みオブジェクトです。

● String オブジェクトのインスタンスは文字列

String オブジェクトは文字列を扱うための組み込みオブジェクトです。String オブジェクトのインスタンスは、new 演算子を使ったコンストラクタで作ります。

▼ 構文：String オブジェクトのインスタンスの作成

```
var 変数名 = new String('文字列');
```

しかし、このような面倒な書き方をしなくても、すでに学んだ文字列リテラルを代入する方法を覚えていれば問題ありません。なぜなら、JavaScriptは、単純データ型の文字列データでも、String オブジェクトのメソッドやプロパティが使えるような親切なしくみになっているからです。

▼ 構文：文字列リテラルの代入方法

```
var 変数名 = '文字列';
```

● String オブジェクトのプロパティとメソッド

String オブジェクトには、文字列を操作するためのメソッドとプロパティが用意されています。String オブジェクトのインスタンス（と文字列リテラルを代入した変数）で使える、おもなプロパティとメソッドです。

▼ おもなプロパティ

プロパティ名	機能
length	文字列の文字数を返す



▼ おもなメソッド

メソッド名	機能
charAt()	指定された位置にある文字を返す
charCodeAt()	指定された位置にある文字の文字コードを返す
concat()	複数の文字列を連結した文字列を返す（元の文字列は変更されない）
indexOf()	指定した文字列を先頭から検索し、先頭文字の位置を返す
lastIndexOf()	指定した文字列を末尾から検索し、先頭文字の位置を返す
match()	正規表現で文字列を検索し、マッチした部分文字列を返す
replace()	正規表現で文字列を検索し、マッチ部分を置換した文字列を返す
search()	正規表現で文字列を検索し、マッチした部分の先頭文字位置を返す
slice()	指定された範囲の文字列を返す（元の文字列は変更されない）
substring()	指定された範囲の文字列を返す（元の文字列は変更されない）
toLowerCase()	すべてのアルファベットを小文字に変換した文字列を返す
toUpperCase()	すべてのアルファベットを大文字に変換した文字列を返す

● replace() メソッドを使ったサンプル

先ほど、RegExp オブジェクトの test() メソッドを使った検索方法を紹介しましたが、String オブジェクトの match() メソッドや、search() メソッドを使っても検索できます。また、replace() メソッドを使えば置換ができます。ここでは、replace() メソッドを使った例を挙げておきます。

```
1 var r = /windows/ig;
2 var notice = 'windows の最新版は WINDOWS 8 です。';
3 notice = notice.replace(r, 'Windows');
4 window.alert(notice);
```

このプログラムは、変数 notice に入っている「windows の最新版は WINDOWS 8 です。」という文字列の「windows」と「WINDOWS」という表記を、「Windows」に置換するものです。1 行目の正規表現では、フラグに i と g を指定しています。

replace() メソッドは、第 1 引数に検索文字列、第 2 引数に置換文字列を指定します (3 行目)。今回は、検索文字列が変数 r、置換文字列が「Windows」なので、それぞれ引数に指定しています。置換後の notice を警告ダイアログボックスに表示して終了します (4 行目)。

Lesson 4

SECTION 7

Number オブジェクト



Number オブジェクトは、数値を使うための組み込みオブジェクトです。数値を操作するためのプロパティやメソッドが用意されています。

● Number オブジェクトのインスタンスは数値

Number オブジェクトのインスタンスも、new 演算子を使ったコンストラクタで作ります。

▼ 構文：Number オブジェクトのインスタンスの作成

```
var 変数名 = new Number(数値);
```

しかし、String オブジェクト同様、数値を扱うときは、すでに学んだ数値リテラルを代入する方法で十分です。単純データ型の数値データでも、Number オブジェクトのメソッドやプロパティが使えるようになっています。

● Number オブジェクトのプロパティとメソッド

Number オブジェクトのインスタンス（と数値リテラルを代入した変数）で使える、おもなプロパティとメソッドです。

▼ Number オブジェクトのおもなプロパティとメソッド

プロパティ・メソッド名	機能
MAX_VALUE	JavaScript で利用できる最大値を返す
MIN_VALUE	JavaScript で利用できる最小値を返す
NaN	「数値ではない」ことを表す値（Not a Number の略）を返す
NEGATIVE_INFINITY	負の無限大を返す
POSITIVE_INFINITY	正の無限大を返す
toExponential()	数値を指数形式の文字列（e を使用）に変換して返す
toFixed()	小数点以下の桁数を指定して、文字列に変換して返す
toString()	数値を文字列に変換して返す



Function オブジェクト

Function オブジェクトは、関数やメソッドを作るための組み込みオブジェクトです。

● コンストラクタを使ってインスタンスを作るには？

コンストラクタを使って関数を定義する方法は、すでに2-3-5で紹介しました。

▼ 構文：コンストラクタを使った関数の定義方法

```
var 変数名 = new Function("仮引数", "処理");
```

このようにコンストラクタと変数を使って定義された関数を**無名関数**といいます。

● Function オブジェクトを使ったサンプル

この方法を使って、2-3で出てきたcircle()関数を定義するとこのようになります。

▼ 特定の日時を表示するサンプルプログラム

```
1 var circle = new Function('r', 'return r * r * 3.14');  
2 window.alert(circle(7));
```

StringオブジェクトやNumberオブジェクト、Arrayオブジェクトと同じで、自分でプログラムを書く際は、基本は関数リテラルを使って定義すれば良いです。

この本では、文法をひととおり知っておいた方が良いという観点から、コンストラクタを使った定義方法も説明しました。

コンストラクタを使った関数の定義を見て、とりあえずそれが関数を定義した文であることと、どのような関数を定義してあるのかがわかれば十分です。



その他の 組み込みオブジェクト

その他のおもな組み込みオブジェクトについて、簡単に説明しておきます。

● Object オブジェクト

Object オブジェクトは、他の組み込みオブジェクトとはちょっと位置づけが異なります。このオブジェクトは、他の組み込みオブジェクトや、プログラマがゼロから作ったオブジェクトに対して、共通して利用できるプロパティやメソッドを定義しているオブジェクトです。言うなれば、オブジェクトのひな形となる組み込みオブジェクトです。

初級者がプログラミングをする際に、Object オブジェクトのインスタンスを作って利用するということはほとんどありません。変数にオブジェクトを代入したいときは、1-6で紹介した方法を使ってください。

● Boolean オブジェクト

Boolean オブジェクトは、独自のプロパティやメソッドを持っていない組み込みオブジェクトです。コンストラクタを使って、論理値をデータとして持つインスタンスを作ることができますが、通常は、1-4-4で紹介したtrueかfalseを変数に代入する方法を使ってください。

● Error オブジェクト

Error オブジェクトは、おもにエラーメッセージを表示するための組み込みオブジェクトです。プログラムにはエラーがつきものです。エラーが出ると、普通はプログラムが停止しますが、プログラミングの際に、エラーが起きた際の処理を、あらかじめ書いておくことができます。これを**例外処理**といいます。

例外処理が書いてあるプログラムでは、プログラムでエラーが発生すると、自動的に Error オブジェクトが作られ、エラーの内容が代入されて、エラー処理を担当する部分に引き渡されます。

エラー処理を担当する部分は、Error オブジェクトの内容を使って、ブラウザにエラー内容を表示します。

この本では、例外処理については説明していませんので、とりあえず、例外処理というものがあるということだけ、覚えておいてください。

Day 3 Lesson 1

window オブジェクトを使ってブラウザを操作する

このレッスンでは、JavaScriptでブラウザを操作するために用意されている window オブジェクトについて学びます。

- 1 window オブジェクト
- 2 alert() メソッドなど
- 3 open() メソッド
- 4 innerWidth プロパティなど
- 5 location オブジェクト
- 6 history オブジェクト
- 7 navigator オブジェクト
- 8 document オブジェクト



window オブジェクト

JavaScript でブラウザを操作するには、window オブジェクトを使います。

● window オブジェクトとは？

JavaScript を使うと、ブラウザそのものをあれこれ操作することができます。ブラウザを操作するには、**window オブジェクト**というオブジェクトを使います。

このwindow オブジェクトは、ブラウザがあらかじめ用意してくれているオブジェクト(ブラウザオブジェクト)なので、プログラマが作る必要はありません。

● window オブジェクトのメソッド

window オブジェクトには、ブラウザに関するデータが入ったプロパティや、ブラウザを操作するメソッドがいろいろと入っています。たとえば、window オブジェクトのメソッドには、以下のようなものがあります。

▼ window オブジェクトのおもなメソッド

メソッド名	機能
alert()	警告ダイアログボックスを表示する
confirm()	確認ダイアログボックスを表示する
prompt()	入力ダイアログボックスを表示する
open()	新しいブラウザウィンドウを開く
close()	開いたブラウザウィンドウを閉じる
print()	印刷用の画面を表示する
scrollBy()	ウィンドウ画面の中身を、指定した分量だけスクロールする
scrollTo()	ウィンドウ画面の中身を、指定した位置までスクロールする



●

window オブジェクトのプロパティ

window オブジェクトのプロパティには、以下のようなものがあります。

▼

window オブジェクトのおもなプロパティ

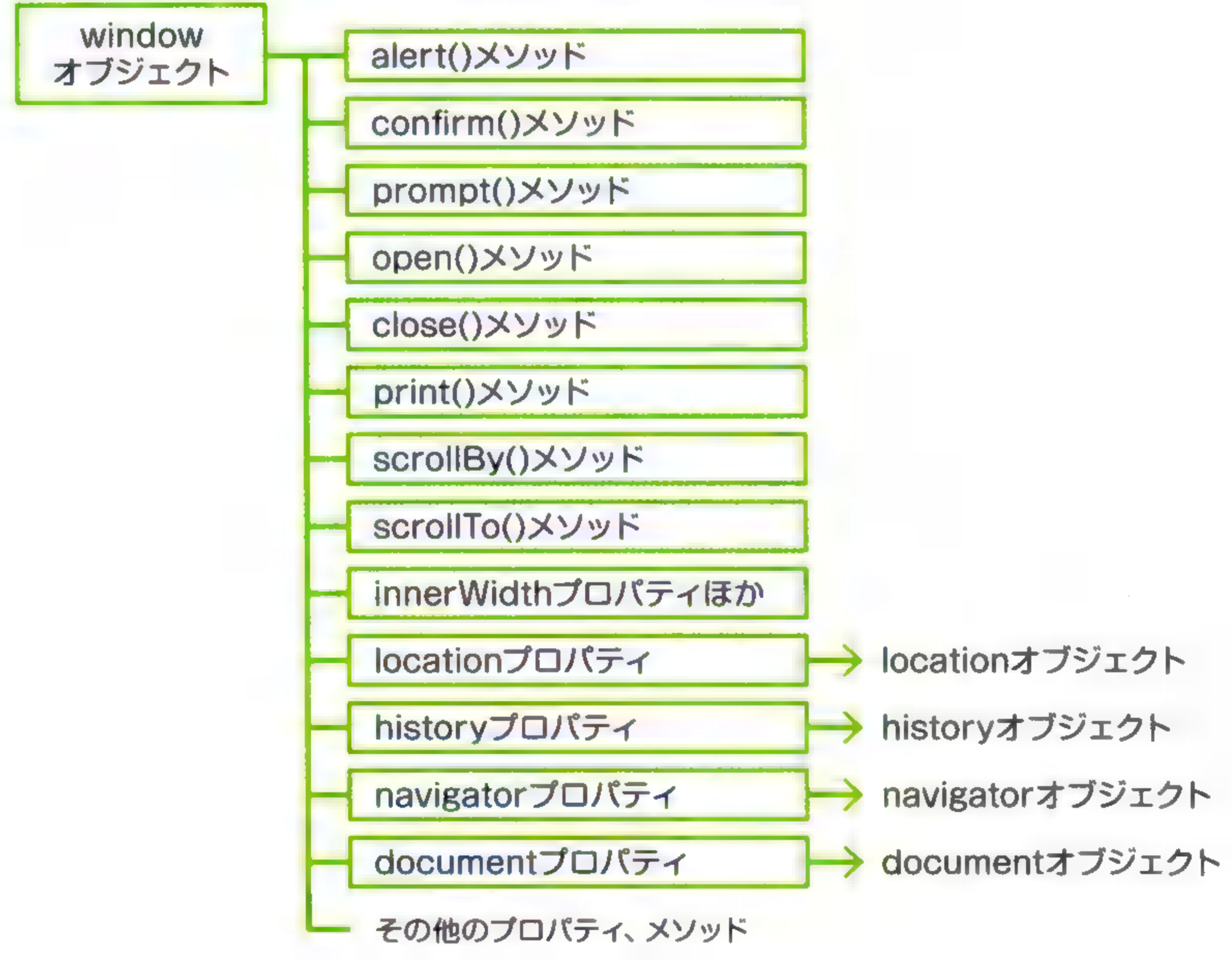
プロパティ名	内容
outerHeight	ブラウザウィンドウの高さを表す数値（単位はピクセル）
outerWidth	ブラウザウィンドウの幅を表す数値（単位はピクセル）
innerHeight	ブラウザウィンドウ画面の高さを表す数値（単位はピクセル）
innerWidth	ブラウザウィンドウ画面の幅を表す数値（単位はピクセル）
location	location オブジェクトが入っている
history	history オブジェクトが入っている
navigator	navigator オブジェクトが入っている
document	document オブジェクトが入っている

outerHeight プロパティ、outerWidth プロパティ、innerHeight プロパティ、innerWidth プロパティには、ブラウザウィンドウに関する数値が入っています。

また、location プロパティ、history プロパティ、navigator プロパティ、document プロパティには、それぞれオブジェクトが入っています。

▼

window オブジェクトの構造



alert() メソッドなど

ダイアログボックス（ポップアップウィンドウ）を表示する alert() メソッド、confirm() メソッド、prompt() メソッドについて説明します。

● alert() メソッドの書き方

alert() メソッドは、これまで何度も使ってきたのでおなじみだと思います。警告ダイアログボックスを表示するメソッドですが、じつは window オブジェクトのメソッドだったのです。

```
window.alert('これは警告ダイアログボックスです');
```

オブジェクトのメソッドやプロパティを使うときには、「オブジェクト名.メソッド名（プロパティ名）」と書くのでした（1-6-2、2-3-5 参照）。

しかし、window オブジェクトは JavaScript では特別なオブジェクトと見なされていて、省略して書くことが許されています。ですので「alert();」と書いても問題なく動作します。

これは他の window オブジェクトのプロパティやメソッドにもあてはまります。この本では、window オブジェクトであることを意識するため、「window.」を付けて書いていますが、通常は省略して書いてかまいません。

● confirm() メソッドと prompt() メソッド

confirm() メソッドは確認ダイアログボックスを、prompt() メソッドは入力ダイアログボックスを開くためのメソッドです。基本的な使い方は、alert() メソッドと同じです。

```
window.confirm('これは確認ダイアログボックスです');  
window.prompt('これは入力ダイアログボックスです');
```


Lesson 1



open() メソッド

新しくウィンドウを開いて、表示内容やサイズをコントロールするには open() メソッドを使います。

● 新しいウィンドウを開くには？

新しいウィンドウを開くには、window オブジェクトの open() メソッドを使います。

▼ 構文：open() メソッド①

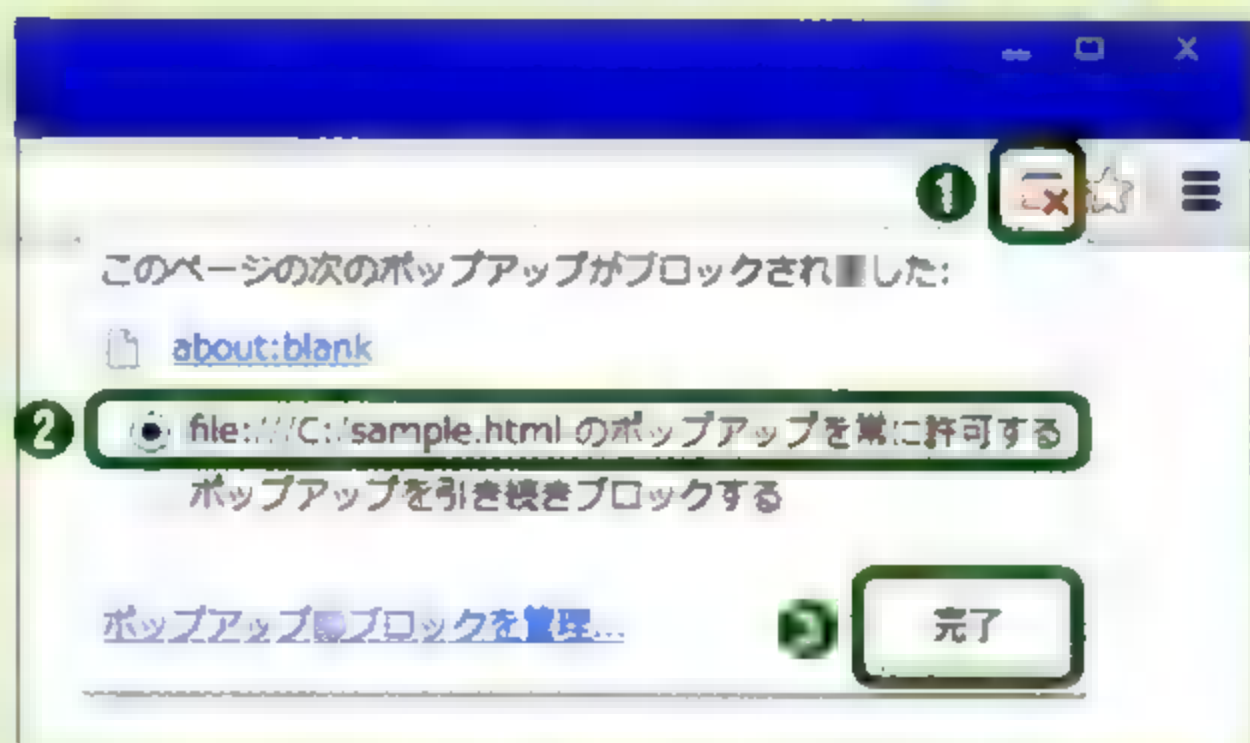
```
window.open()
```

たとえば、このようなソースコードを実行すると、新しいウィンドウが開きます。

```
window.open();
```

MEMO

Chrome で open() メソッドを使って新しいウィンドウ（ポップアップ）を開こうとすると、危険なサイトの悪用を防ぐため、あらかじめブロックされるような設定になっています。もし、作ったサンプルプログラムがブロックされたときは、以下の手順を行ってください。



- ① アイコンをクリック
- ② 「～のポップアップを常に許可する」をチェック
- ③ 「完了」をクリック
- ④ HTML ファイルをリロード

● URL を指定して新しいウィンドウを開くには？

URL を指定して新しいウィンドウを開きたいときには、`open()` メソッドの引数に URL を指定します。

▼ 構文：`open()` メソッド②

```
window.open('URL', 'ウィンドウ名')
```

たとえば、このようなソースコードを実行すると、ソシムのWebページ (<http://www.socym.co.jp/>) が表示された新しいウィンドウが開きます。

```
window.open('http://www.socym.co.jp/', 'socym');
```

MEMO

引数として指定する URL には、`http://` (または `https://`) も忘れずに付けるようにしてください。
「ウィンドウ名」には、好きな名前を付けてかまいません。

● サイズを指定して新しいウィンドウを開くには？

開くウィンドウのサイズを調整したいときには、`open()` メソッドの3番目の引数に、ウィンドウのサイズを指定します。

▼ 構文：`open()` メソッド③

```
window.open('URL', 'ウィンドウ名', 'width=幅,height=高さ')
```

たとえば、幅600ピクセル、高さ450ピクセルの新しいウィンドウを開きたいときは、このように書きます。

```
1 window.open('http://www.socym.co.jp/', 'socym',  
  'width=600,height=450');
```

3番目の引数には、いっさい空白 (半角スペースやタブ、改行) を含めてはいけません。エラーになります。

Lesson 1



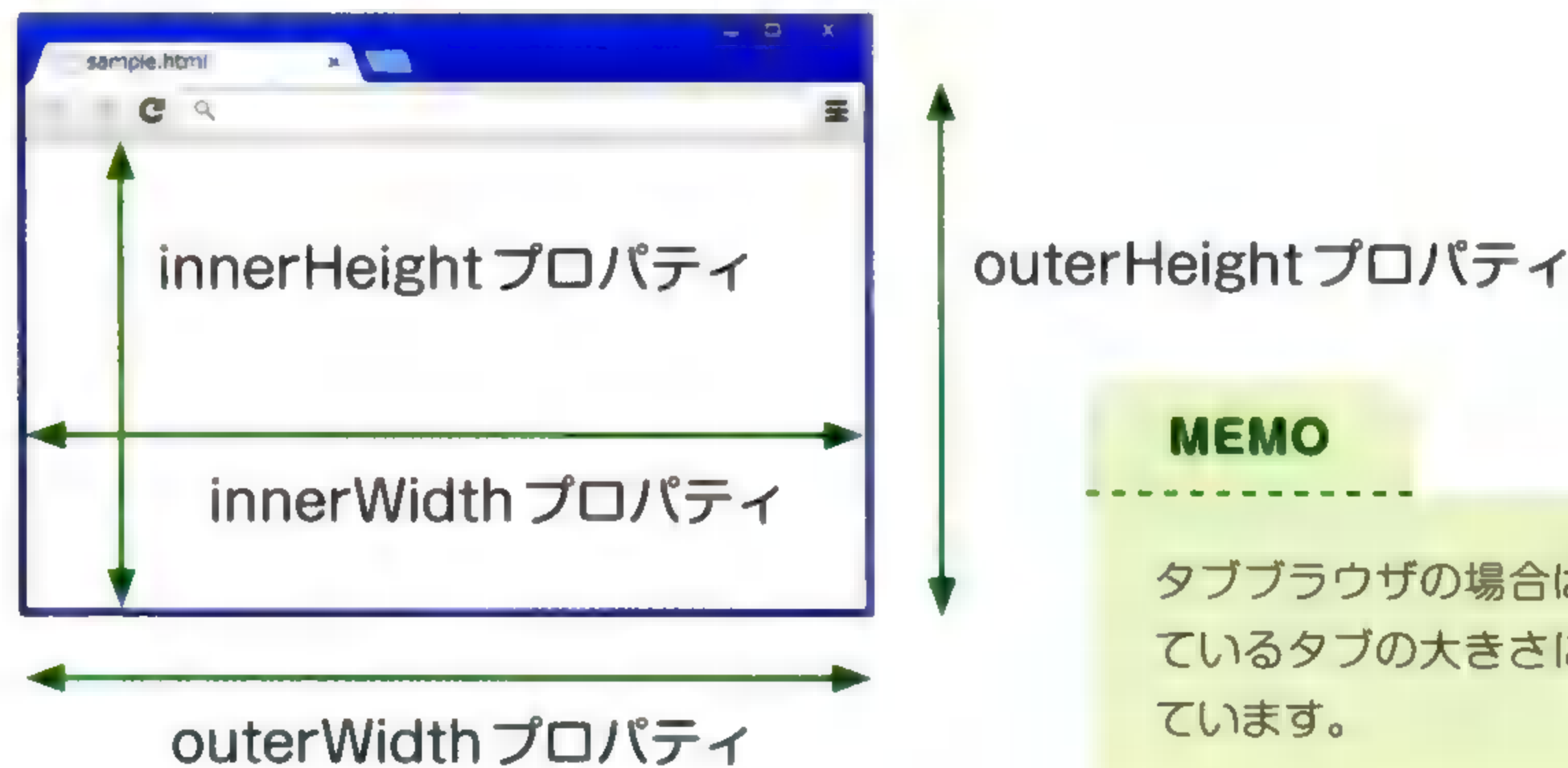
innerHTML プロパティなど

ブラウザウィンドウの大きさを操作するには、innerHTML プロパティなどを使います。

● ウィンドウの数値が入っているプロパティ

outerHeight プロパティ、outerWidth プロパティ、innerHeight プロパティ、innerHTML プロパティには、それぞれブラウザウィンドウに関する数値が入っています。

▼ プロパティとブラウザウィンドウ数値の対応



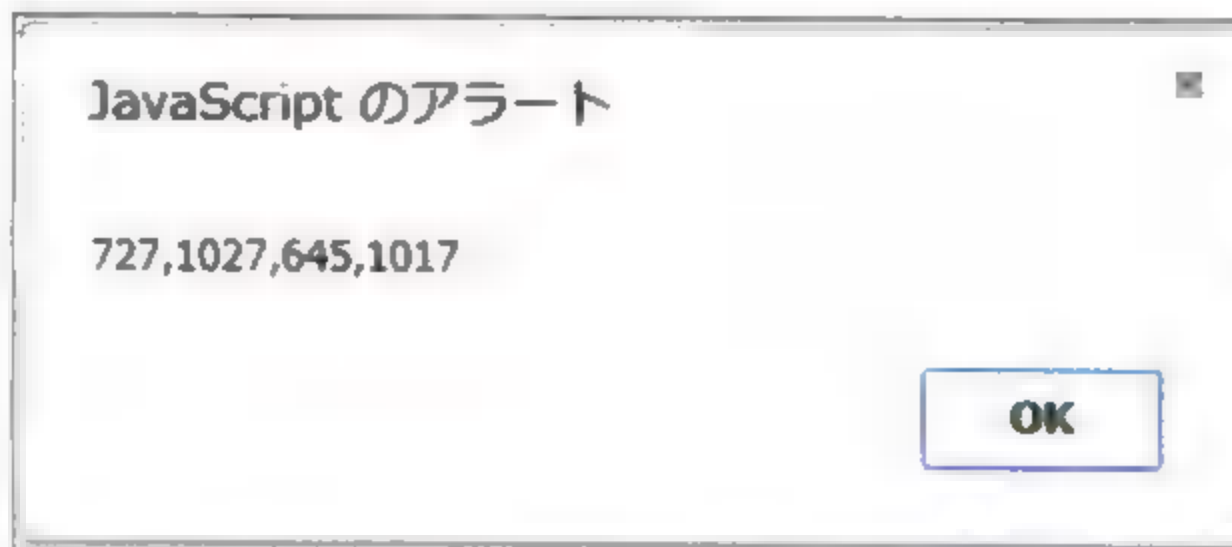
MEMO

タブブラウザの場合は、ファイルを開いているタブの大きさに関する数値が入っています。

● プロパティでブラウザウィンドウに関する値を取得するには？

各プロパティで、開いたウィンドウに関する数値を取得して、警告ダイアログダイアログボックスに表示してみましょう。以下のソースコードを入力して、実行してみてください。

```
1 var oh = window.outerHeight;  
2 var ow = window.outerWidth;  
3 var ih = window.innerHeight;  
4 var iw = window.innerWidth;  
5 alert(oh + ',' + ow + ',' + ih + ',' + iw);
```



ウィンドウ(タブ)の
大きさが表示された

location オブジェクト



ここからは window オブジェクトの中にあるオブジェクトについて順番に説明していきます。location オブジェクトは、Web ページの、URL に関するデータが格納されているオブジェクトです。

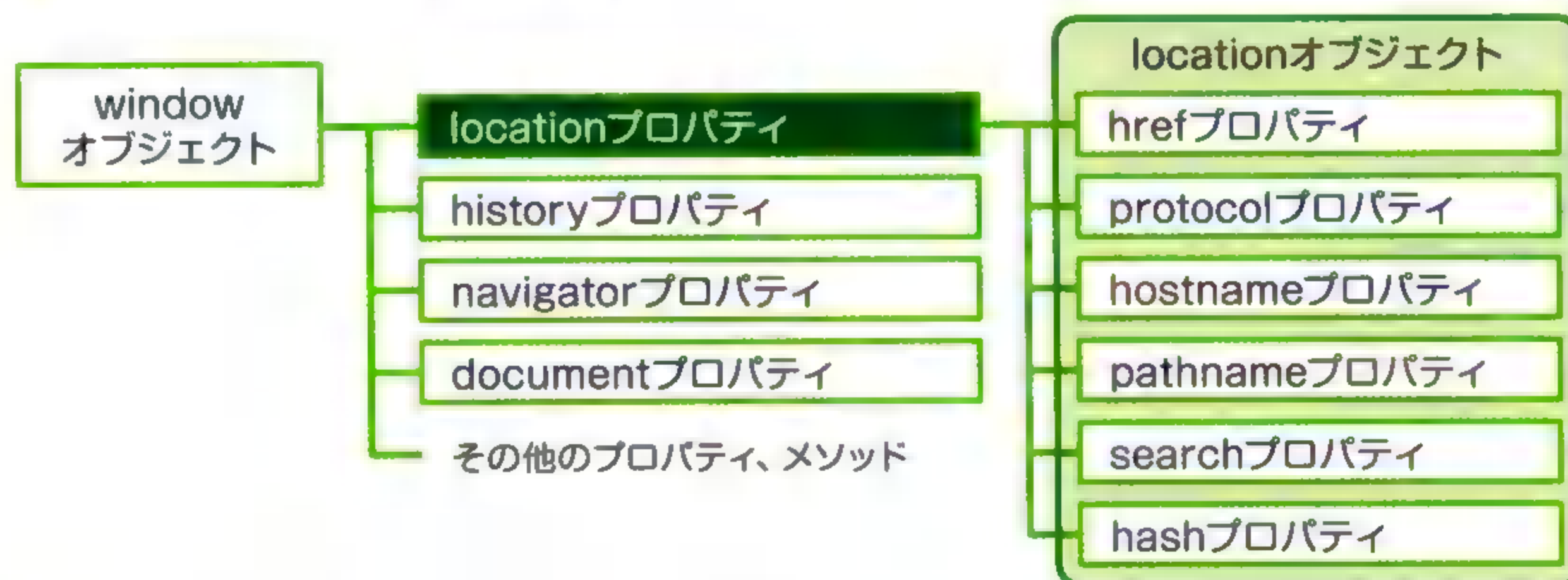


● location オブジェクトとは？

location オブジェクトは、window オブジェクトの location プロパティの中に入っているオブジェクトです。location オブジェクトのプロパティを操作すると、URL に関する情報を取得したり、変更したりすることができます。

location オブジェクトのプロパティには、以下のようなものがあります。

▼ location オブジェクトの構造



▼ location オブジェクトのおもなプロパティ

プロパティ名	内容	例
href	URL 全体	http://www.socym.co.jp/?cat=1&target=normal&s=html#pageTtl
protocol	プロトコル	http:
hostname	ドメイン名	socym.co.jp
pathname	ファイルまでのパス	/
search	クエリ文字列	?cat=1&target=normal&s=html#pageTtl
hash	ハッシュ値	#pageTtl

これらのプロパティのうち、よく使われるプロパティは href プロパティです。



● href プロパティで URL の値を取得するには？

href プロパティには、表示されている Web ページの URL が入っています。たとえば、href プロパティに入っている値 (URL) を、警告ダイアログボックスに表示するには、このように書きます。

```
window.alert(window.location.href);
```

sample.html はパソコンの中に保存しているファイルなので、このソースコードを実行すると、URL ではなく、ファイルの置き場所までのパスが表示されますが、このファイルを Web サーバに置いて実行すれば、ファイルの URL が表示されます。

● href プロパティの値を書き換えるとどうなるか？

href プロパティの中の値 (URL) を書き換えると、書き換えたページへ移動します。たとえば、URL を 'http://www.socym.co.jp/' に書き換えてみましょう。

```
window.location.href= 'http://www.socym.co.jp/';
```

このソースコードが書かれた sample.html を実行すると、ページがブラウザで開かれた瞬間、http://www.socym.co.jp/ のページに移動します。



ファイルが開かれると同時に
指定したページへ移動する

href プロパティは「現在開かれているページの URL」を示す値なので、これを別の URL に書き換えると、強制的にそのページを開くように動作します。

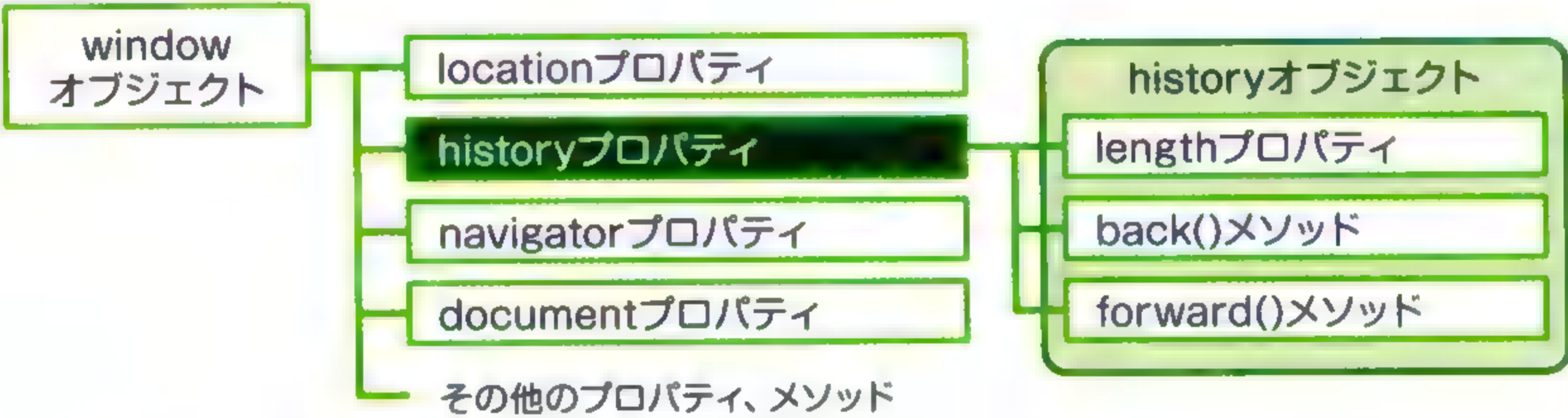
history オブジェクト

history オブジェクトは、Web ページの閲覧履歴に関するデータを管理・操作するオブジェクトです。

history オブジェクトとは？

history オブジェクトは、window オブジェクトの history プロパティの中に入っているオブジェクトです。history オブジェクトのプロパティやメソッドを使うと、閲覧履歴件数を取得したり、ページを戻したり、先に進めたりすることができます。history オブジェクトのプロパティとメソッドには、以下のようなものがあります。

history オブジェクトの構造



history オブジェクトのおもなプロパティとメソッド

プロパティ名 (メソッド名)	内容・機能
length プロパティ	閲覧履歴の件数
back() メソッド	1 つ前のページに戻る
forward() メソッド	1 つ後のページへ進む

MEMO

ユーザーのプライバシーを侵害することを防ぐため、history オブジェクトには、閲覧履歴の URL を取得するするようなプロパティやメソッドはありません。



navigator オブジェクト

SECTION

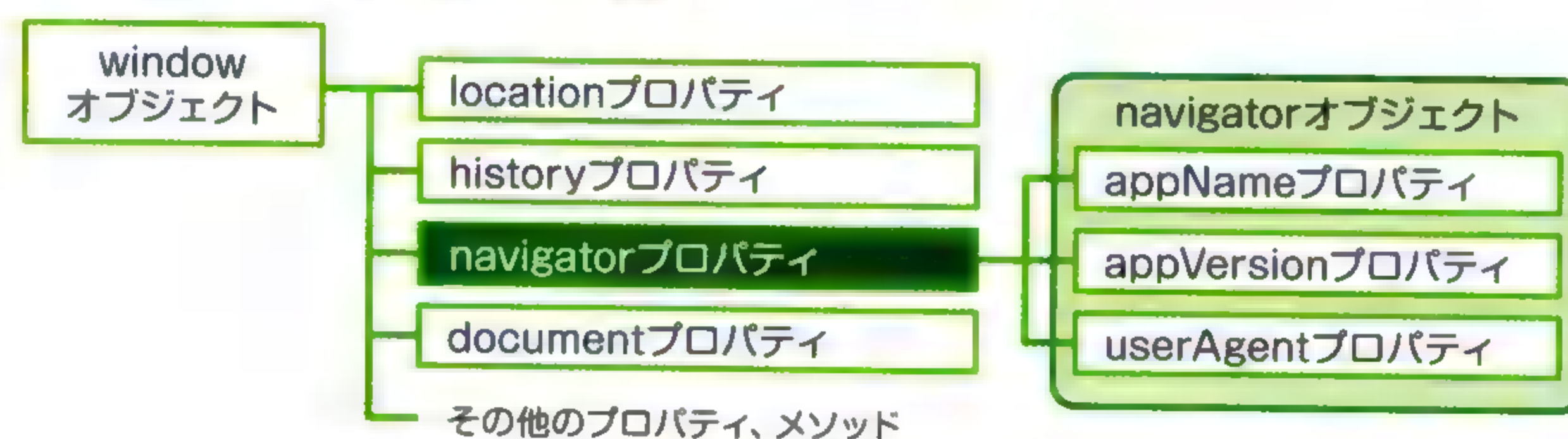
7

navigator オブジェクトは、ブラウザの種類やバージョンに関するデータを管理しているオブジェクトです。

● navigator オブジェクトとは？

navigator オブジェクトは、window オブジェクトの navigator プロパティの中に入っているオブジェクトです。ブラウザの種類やバージョンに関するデータが入ったプロパティが用意されています。navigator オブジェクトのプロパティには、以下のようなものがあります。

▼ navigator オブジェクトの構造



▼ navigator オブジェクトのおもなプロパティ

プロパティ名	内容
appName	ブラウザの名前
appVersion	ブラウザのバージョン
userAgent	ブラウザの情報。正確にはブラウザが HTTP リクエストの user-header に使う値

以下のように書けば、警告ダイアログボックスにブラウザの情報が表示されます。

```
window.alert(window.navigator.userAgent);
```

userAgent プロパティの値は、たとえば、ブラウザによって、Web ページの表示内容やデザインを変えたいときの判定に使われます。

Lesson 1



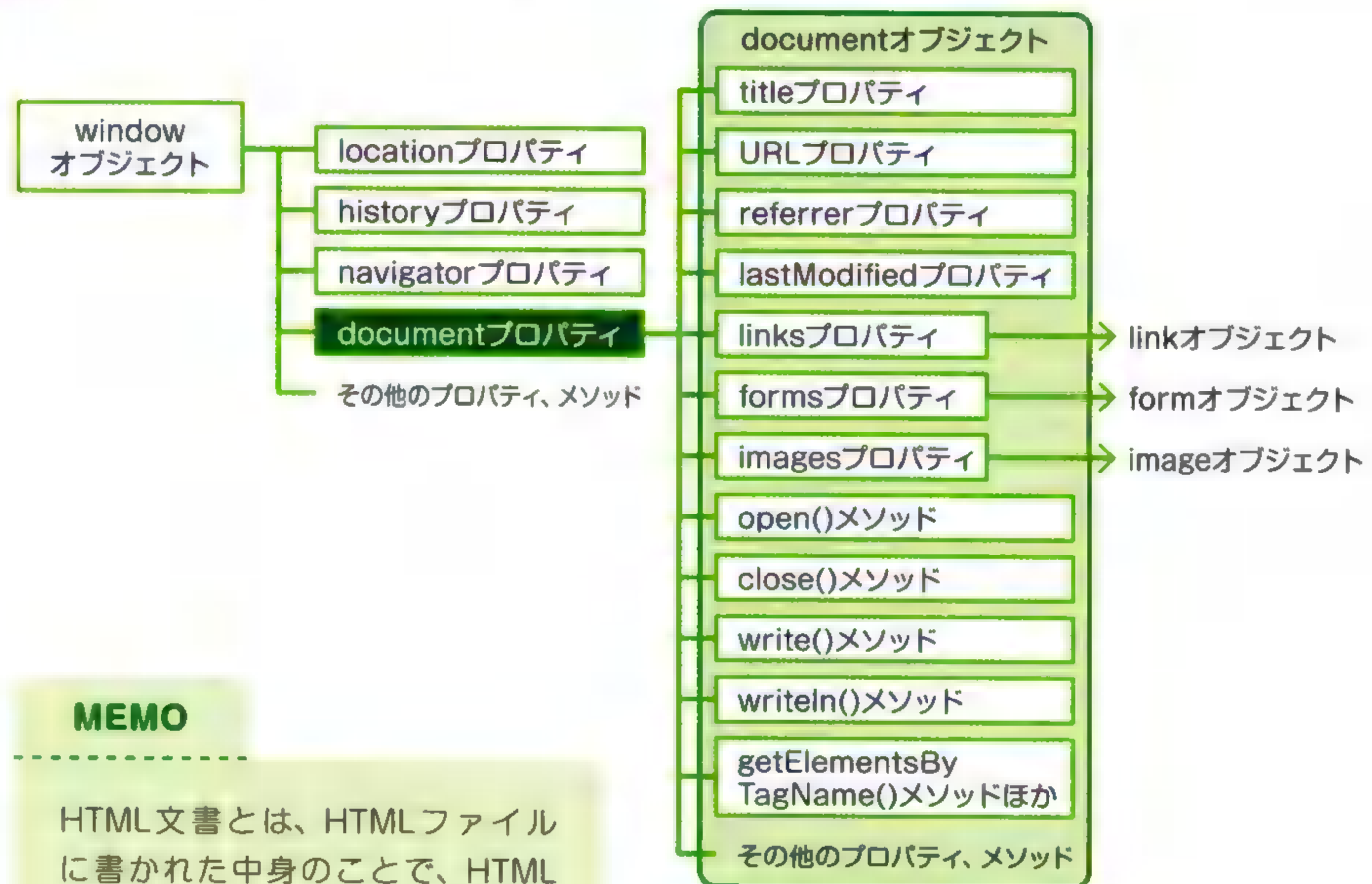
document オブジェクト

document オブジェクトは、ブラウザに読み込まれている HTML 文書に関するデータが入っているオブジェクトで、window オブジェクトの中でもっとも重要なオブジェクトです。

● document オブジェクトとは？

document オブジェクトは、window オブジェクトの document プロパティの中にあるオブジェクトです。ブラウザで表示されている HTML 文書や CSS の内容を取得したり操作したりするために使うプロパティやメソッドが用意されています。document オブジェクトのプロパティやメソッドには、以下のようなものがあります。

document オブジェクトの構造



MEMO

HTML 文書とは、HTML ファイルに書かれた中身のことで、HTML ドキュメントともいいます。文書（ドキュメント）内容を扱うので「document」オブジェクトです。

▼ document オブジェクトのおもなプロパティ

プロパティ名	内容
title	HTML 文書の title 要素の内容
URL	HTML 文書の URL
referrer	HTML 文書へリンクしている文書の URL
lastModified	HTML 文書の最終更新日
links	link オブジェクト（文書内のすべての a 要素と area 要素）
forms	form オブジェクト（文書内のすべての form 要素）
images	image オブジェクト（文書内のすべての img 要素）

▼ document オブジェクトのおもなメソッド

メソッド名	機能
open()	新しい文書の出力を開始する
close()	新しい文書の出力を終了する
write()	文書内に文字列を出力する
writeln()	文書内に文字列を出力して、改行する
getElementsByTagName()	要素を取得する
getElementById()	id 名にマッチする要素を取得する
getElementsByClassName()	class 名にマッチする要素を取得する
querySelector()	CSS のセレクタ名にマッチする要素を取得する

COLUMN

link オブジェクト、form オブジェクト、image オブジェクト

link オブジェクトは、HTML 文書のリンクに関する情報を取得したり、操作したりするためのオブジェクトです。link オブジェクトの実体は、links という名前の配列です。その要素には、文書内のすべてのリンク（a 要素）のデータがオブジェクトとして入っています。たとえば、HTML 文書内の上から2番目のa要素のURL値を取得するには、href プロパティを使って、「document.links[1].href」と書きます。

form オブジェクトは、HTML 文書のフォームに関する情報を取得したり、操作したりするためのオブジェクトです。form オブジェクトもまた、実体は forms という名前の配列ですので、要素には、文書内のすべてのフォーム（form 要素）のデータがオブジェクトとしてしまわれています。

image オブジェクトは、HTML 文書の画像に関する情報を取得したり、操作したりするためのオブジェクトです。これもまた実体は images という名前の配列です。配列の要素には、文書内のすべての画像（img 要素）のデータがオブジェクトとして入っています。

Day 3 Lesson 2

DOM の基本的な使い方

このレッスンでは、JavaScriptでHTML文書の中身をいろいろと操作するためのしくみである、DOMの基本的な使い方を学びます。

- 1 DOMとは何か？
- 2 要素名で要素を取得する
- 3 要素の内容を取得する
- 4 要素の内容を変更する
- 5 id属性の値にマッチする要素を取得する
- 6 class属性の値にマッチする要素を取得する





DOM とは何か？

JavaScript で HTML 文書をデータとして扱うためのしくみである DOM について学びます。

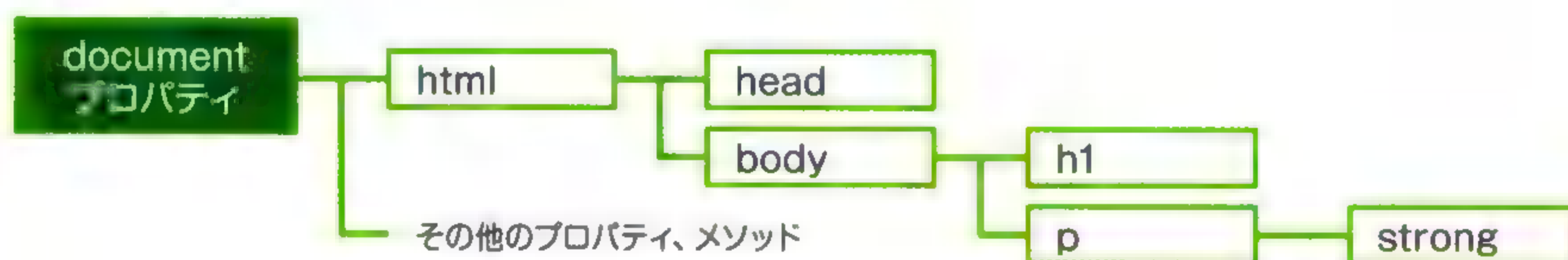
● DOM とは何か？

DOM (Document Object Model) とは、HTML 文書をオブジェクトとして読み込んで操作するためのしくみです。異なるブラウザでも等しく動作するように仕様が定められています。DOM を使うと、HTML 文書や CSS の内容を、オブジェクトを扱うのと同じ方法で操作することができます。

たとえば、以下のような HTML 文書があります。

```
1 <html lang="ja">
2 <head>
3   <title> プログラム実行テスト </title>
4 </head>
5 <body>
6   <h1> プログラム実行テスト </h1>
7   <p> 現在は <strong>3 日目のレッスン 1</strong> です。 </p>
8 </body>
9 </html>
```

この HTML 文書は、document プロパティの中に、次のような構造のオブジェクトとして読み込まれます。



しかし、要素名をそのままオブジェクト名やプロパティ名としては使えません。要素を操作するには、document オブジェクトのメソッドを使います。



要素名で要素を取得する

読み込まれた HTML 文書の中から、要素名（タグ名）で要素を検索して、データを取得するメソッドの使い方を説明します。



● HTML文書の要素をデータとして取得するには？

HTML 文書の要素をデータとして取得するには、document オブジェクトにあらかじめ用意されている `getElementsByTagName()` メソッドを使います。

▼ 構文：`getElementsByTagName()` メソッド

`document.getElementsByTagName('取得したい要素名')`

取得したい要素名を、メソッドの引数に指定すれば OK です。

大文字と小文字の違いに注意してください。「get」はすべて小文字ですが、「Elements」「By」「Tag」「Name」の1文字目はそれぞれ大文字です。また、要素という意味の「Elements」には複数形を表す「s」が付いています。忘れないようにしましょう。

● p 要素を取得するサンプルプログラム

`getElementsByTagName()` メソッドを使ったサンプルプログラムを作ってみましょう。p 要素をすべて取得して変数 ps に代入し、最初の p 要素を警告ダイアログボックスに表示するプログラムです。ソースコードを入力して、実行してみてください。

```
1 <body>
2   <h1> プログラム実行テスト </h1>
3   <p> 第 1 段落 </p>
4   <p> 第 2 段落 </p>
5   <p> 第 3 段落 </p>
```

MEMO

body 要素以外は省略しています。

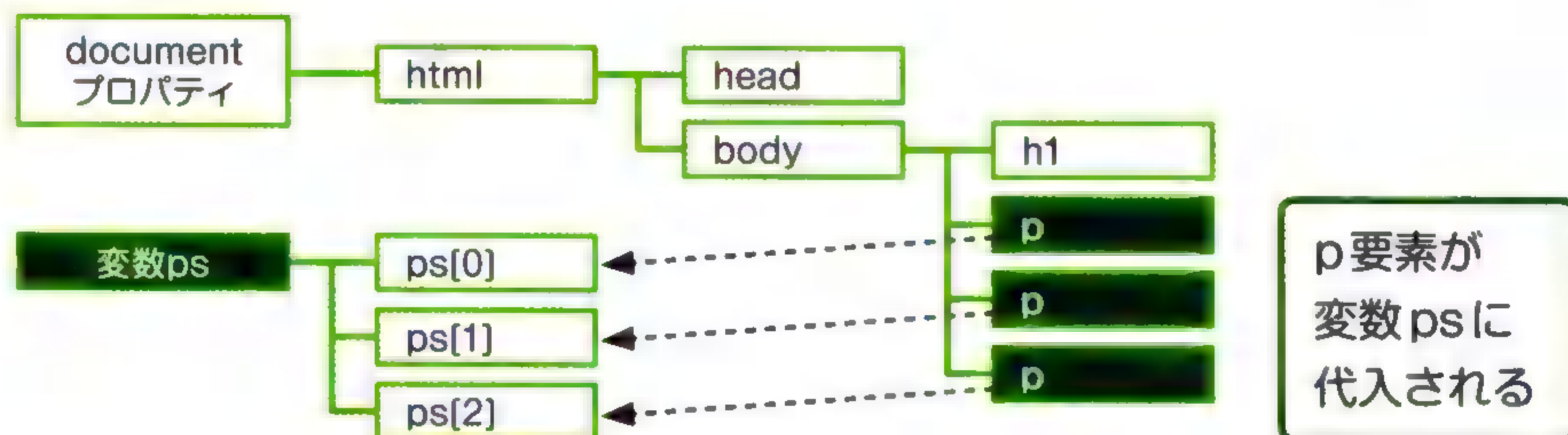



```

6      <script>
7          var ps = document.getElementsByTagName('p');
8          alert(ps[0]);
9      </script>
10     </body>

```

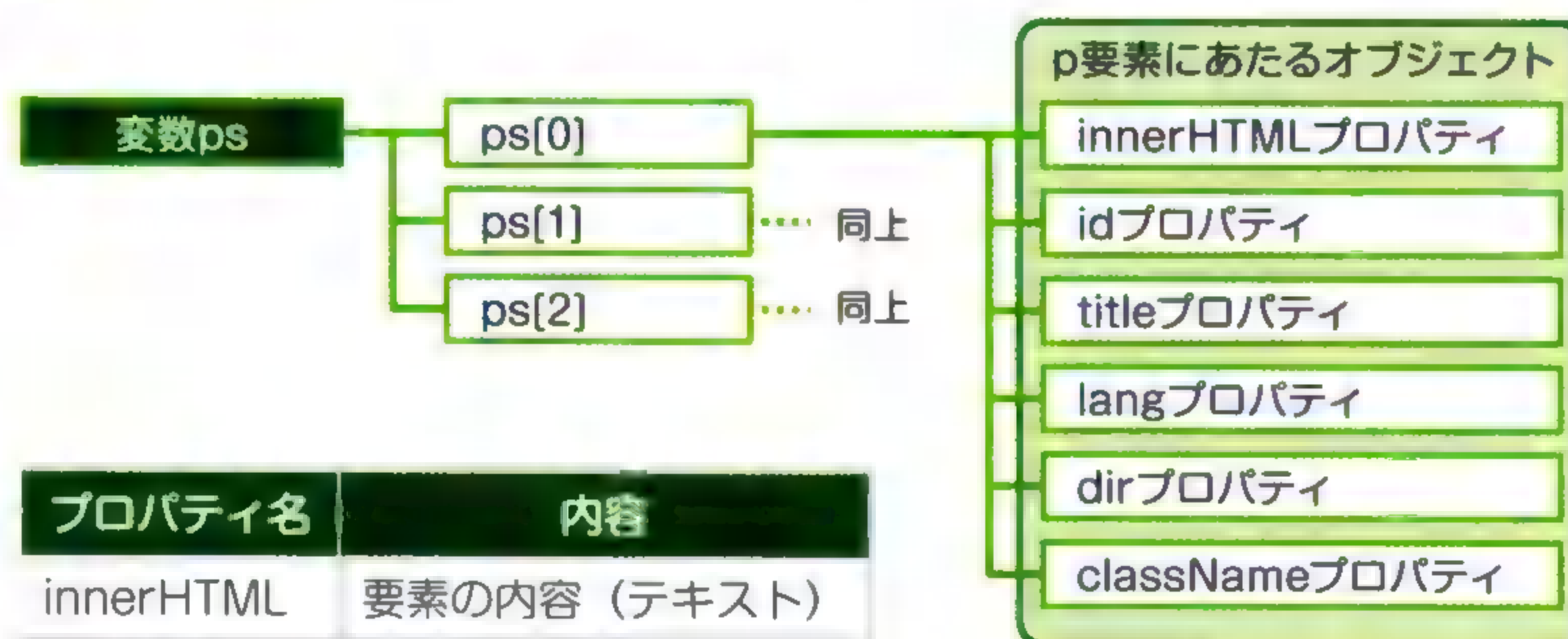
このプログラムを実行すると、3つあるp要素がそれぞれ、配列の要素のような形として、変数psに代入されます(7行目)。取得する順番は、先頭から順番になるので、ps[0]に最初のp要素、ps[1]に2番目のp要素、ps[2]に3番目のp要素が代入されます。



代入されたp要素はオブジェクトになっています。オブジェクトなので、プロパティやメソッドが用意されています。

このp要素オブジェクトには、次のようなプロパティが用意されていて、それぞれp要素の該当するデータが代入されています。

▼ p要素オブジェクトのおもなプロパティ



Lesson 2

SECTION 3

要素の内容を取得する

要素の内容をデータとして取得するには、innerHTML プロパティを使います。

● p 要素の内容をデータとして取得するには？

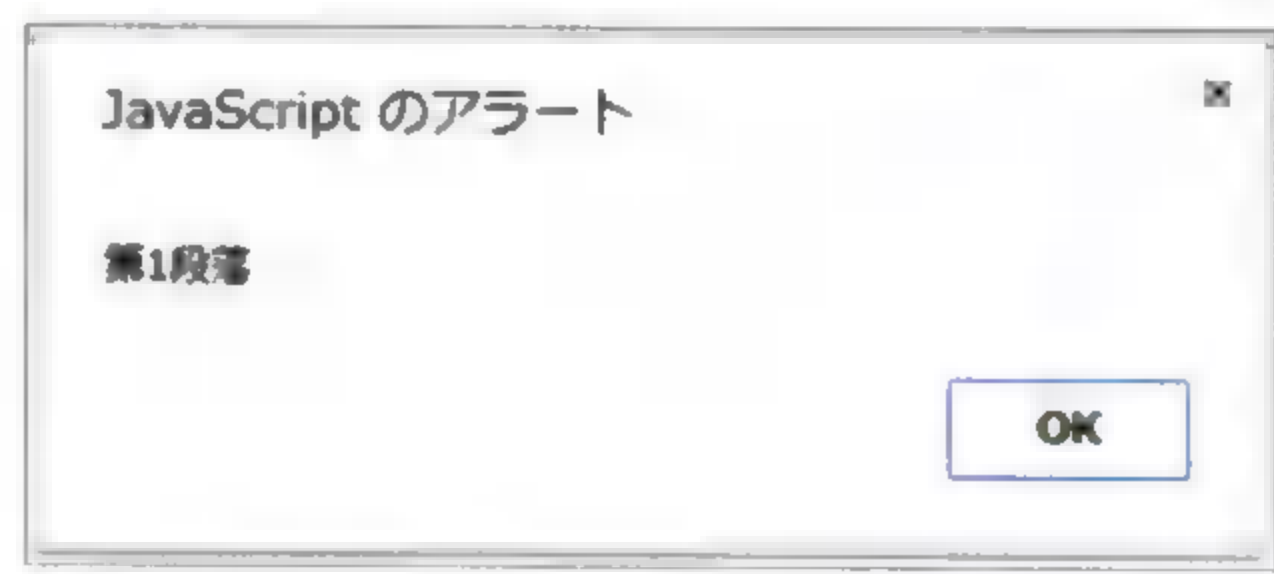
取得した要素の内容をデータとして取得するには、innerHTML プロパティを使います。

innerHTML プロパティの参照方法

オブジェクト名 .innerHTML

たとえば、ps[0]にオブジェクトとして代入されているp要素の内容を取得して、警告ダイアログボックスに表示するには、次のように書きます。

```
7 var ps = document.getElementsByTagName('p');
8 alert(ps[0].innerHTML);
```



最初のp要素の内容が表示される

innerHTML プロパティで取得した要素の内容は、文字列として取得されます。たとえば、要素の内容が「2 + 3」という数値や演算子であっても、計算されずにそのまま「2 + 3」という文字列として表示されます。

また、innerHTML プロパティで取得した要素の内容に別の要素（たとえばb要素）が含まれる場合は、そのタグも文字列として取得されます。たとえば、p要素の内容が「第2要素」という内容であれば、「第2要素」という文字列として表示されます。

要素の内容を変更する

innerHTML プロパティで取得した要素の内容を変更すると、ブラウザでの表示も変更されます。

● p 要素の内容を変更するには？

取得したp要素の内容を変更するには、先ほどと同じく **innerHTML プロパティ** を使います。innerHTML プロパティに文字列を代入すれば、要素の内容は代入した文字列に変更されます。使い方は、これまでに学んだ変数やプロパティと同じです。

▼ innerHTML プロパティへの代入方法

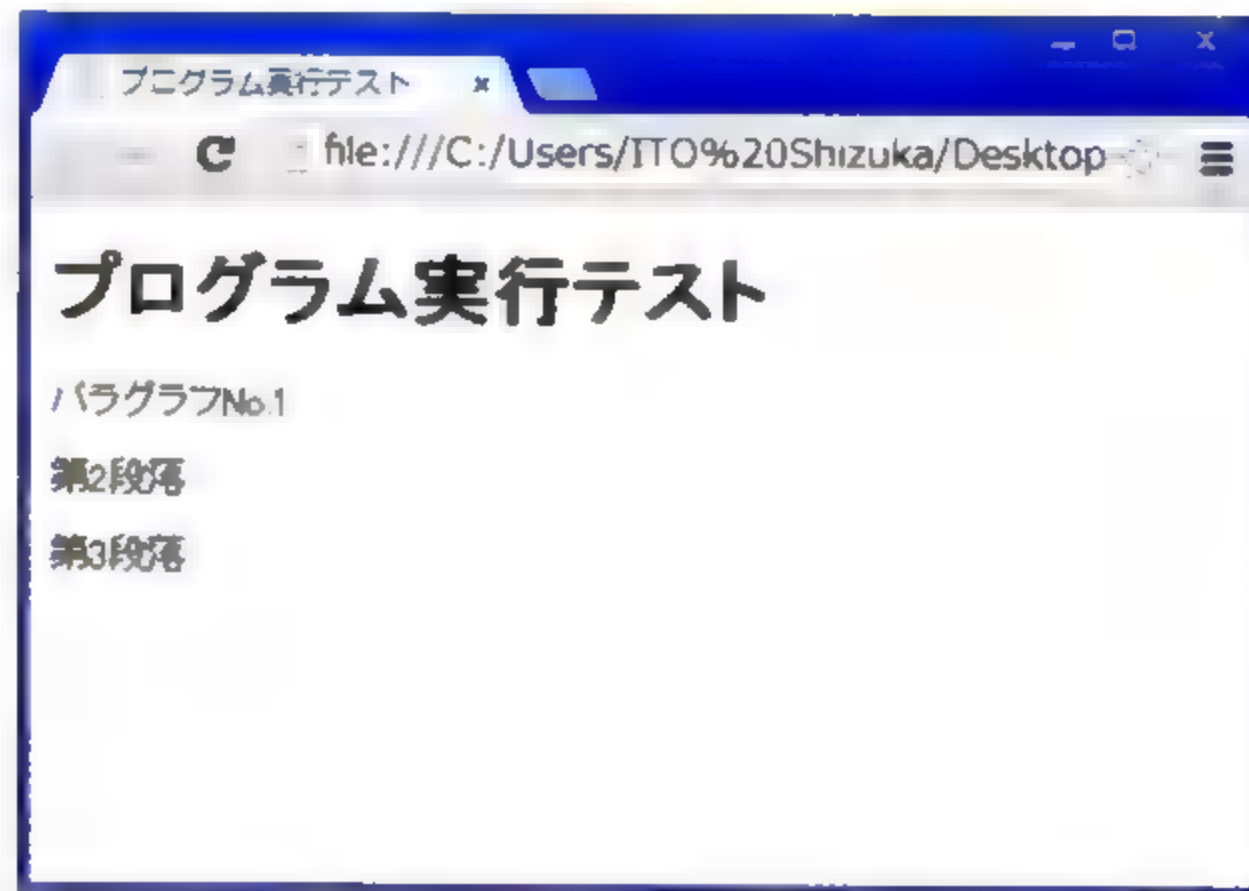
オブジェクト名.innerHTML = '文字列';

たとえば、ps[0]に入っているp要素のinnerHTML プロパティを「パラグラフ No.1」に変更してみましょう。

```
6 <script>
7 var ps = document.getElementsByTagName('p');
8 ps[0].innerHTML = 'パラグラフ No.1';
9 </script>
```

innerHTML プロパティの値を変更すると、もともとの要素の内容が上書きされるので、ブラウザでの表示も変更後の内容になります。

今回の例では、「第1段落」という文字列が、「パラグラフ No.1」という文字列に変わって、ブラウザに表示されます。



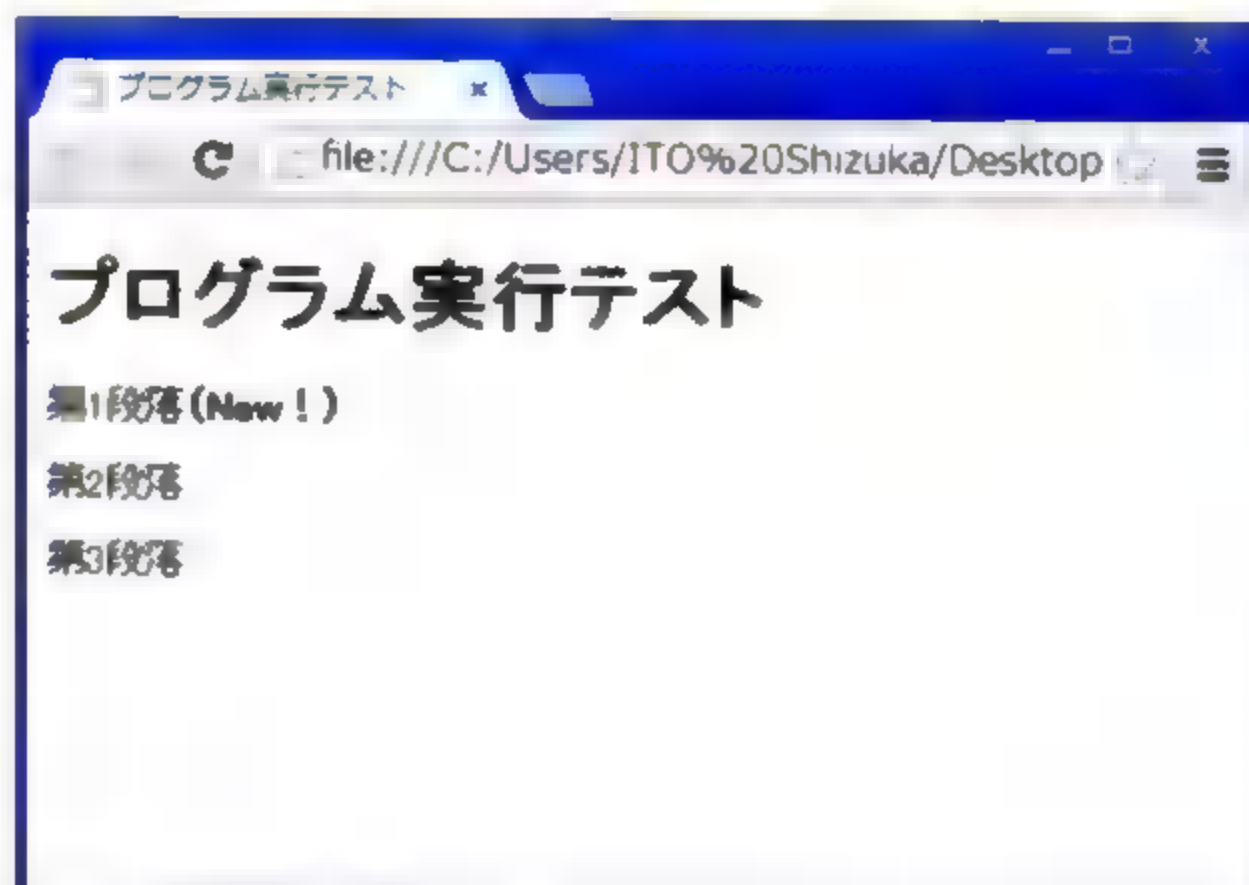
最初のp要素の内容が
変更された

● p 要素の内容に文字列を追加したいときは？

innerHTML プロパティを使えば、p 要素の内容を上書きせずに、元の内容に文字列を付け加える、ということも可能です。たとえば、「第1段落」のあとに「(New!)」という文字列をb要素にして付け加えたい場合は、このように書きます。

```
6 <script>
7 var ps = document.getElementsByTagName('p');
8 ps[1].innerHTML = ps[1].innerHTML + '<b> (New !)</b>';
9 </script>
```

プログラムを実行すると、「第1段落」という文字列のあとに「(New!)」という文字列が追加されて、ブラウザに表示されます。



最初のp要素の内容に
文字列が追加された





id 属性の値にマッチする要素を取得する

読み込まれた HTML 文書の中から、要素の id 属性の値で要素を検索して、データを取得することもできます。

●要素を id 属性の値で検索するには？

id 属性の値で検索して、マッチする要素をデータとして取得するには、document オブジェクトの **getElementById()** メソッドを使います。

▼ 構文：getElementById() メソッド

```
document.getElementById('id属性の値')
```

取得したい要素名の id 属性の値を、メソッドの引数に指定すれば OK です。

MEMO

getElementsByName() メソッドと違って、getElementById() メソッドの「Element」は単数形です。これは、1つのHTML文書には、id属性の値が同じ要素は存在しない、つまり当てはまる要素 (element) は1つだけだからです。

● id 属性値で要素を取得するサンプルプログラム

getElementById() メソッドを使ったサンプルプログラムを作ってみましょう。id 属性の値が bottom である要素を取得して変数 i に代入し、内容を警告ダイアログボックスに表示するプログラムです。ソースコードを入力して、実行してみてください。


```

1  <body>
2    <h1> プログラム実行テスト </h1>
3    <p id="top"> 第1 段落 </p>
4    <p class="contents"> 第2 段落 </p>
5    <p id="bottom"> 第3 段落 </p>
6    <script>
7      var i = document.getElementById('bottom');
8      alert(i.innerHTML);
9    </script>
10 </body>

```

このプログラムを実行すると、id属性の値が「bottom」である要素、つまり3番目のp要素が、変数iにオブジェクトとして代入されます(7行目)。そのオブジェクトのinnerHTMLプロパティの値が、警告ダイアログボックスに表示されます(8行目)。



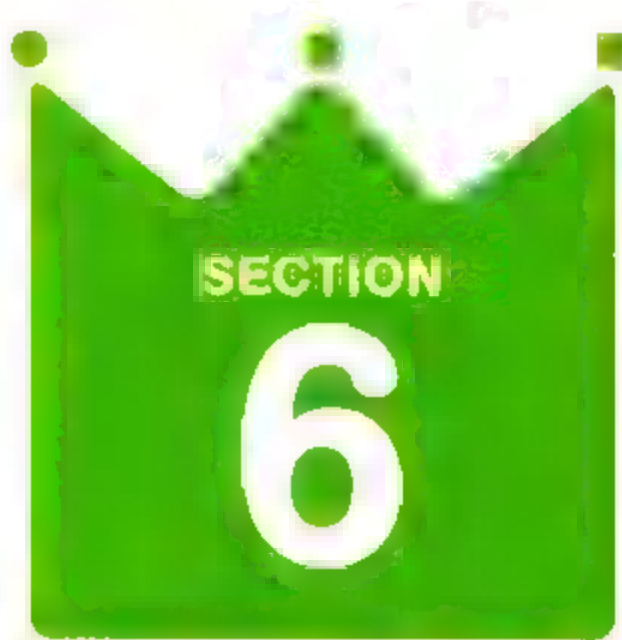
id属性値が「bottom」である
3番目のp要素が取得された

COLUMN

DOM で使えるその他のメソッド

このレッスンで紹介したほかにもメソッドはあります。おもなメソッドを挙げておきます。

メソッドの書き方と引数	機能
document.querySelector('セクタ')	セクタ (CSS) にマッチする要素を1つ取得する
document.querySelectorAll('セクタ')	セクタにマッチする要素をすべて取得する
要素.getAttribute('属性')	要素の属性値を取得する
要素.setAttribute('属性', '属性値1')	属性値を属性値1に変更する
document.createElement('要素')	新しい要素を作る
要素1.appendChild('要素2')	要素2を要素1の子要素として追加する



class 属性の値にマッチする要素を取得する

読み込まれた HTML 文書の中から、要素の class 属性の値で要素を検索して、データを取得することもできます。

● 要素を class 属性の値で検索するには？

class 属性の値で検索して、マッチする要素をデータとして取得するには、**getElementsByClassName() メソッド**を使います。

▼ 構文：getElementsByClassName() メソッド

```
document.getElementsByClassName('class 属性の値')
```

取得したい要素名の class 属性の値を、メソッドの引数に指定します。

● class 属性値で要素を取得するサンプルプログラム

getElementsByClassName() メソッドを使ったサンプルプログラムを作ってみましょう。前のセクションで作ったサンプルプログラムの7行目と8行目を以下のように変えて、実行してみてください。

```
7      var c = document.getElementsByClassName('contents');  
8      alert(c[0].innerHTML);
```

実行して、「第2段落」と表示されれば成功です。このプログラムを実行すると、class 属性の値が「contents」である要素が、配列の要素のような形として変数cに代入されます（7行目）。そのオブジェクトのc[0]のinnerHTMLプロパティの値が、警告ダイアログボックスに表示されます（8行目）。同じclass属性値を持つ要素は、1つのHTML文書の中に複数存在する可能性があるので、変数への代入のされ方は、getElementsByTagName()メソッドと同じになります。

Day 3 Lesson 3

イベントと イベントハンドラ

このレッスンでは、JavaScriptでよく使われる、イベントとイベントハンドラの基本的な使い方を学びます。

- 1 イベントとは何か？
- 2 イベントハンドラの使い方
- 3 開始タグの中で関連付ける方法
- 4 プロパティを使って関連付ける方法
- 5 load イベントの使い方
- 6 `addEventListener()` メソッド
- 7 タッチデバイスのイベント

イベントとは何か？

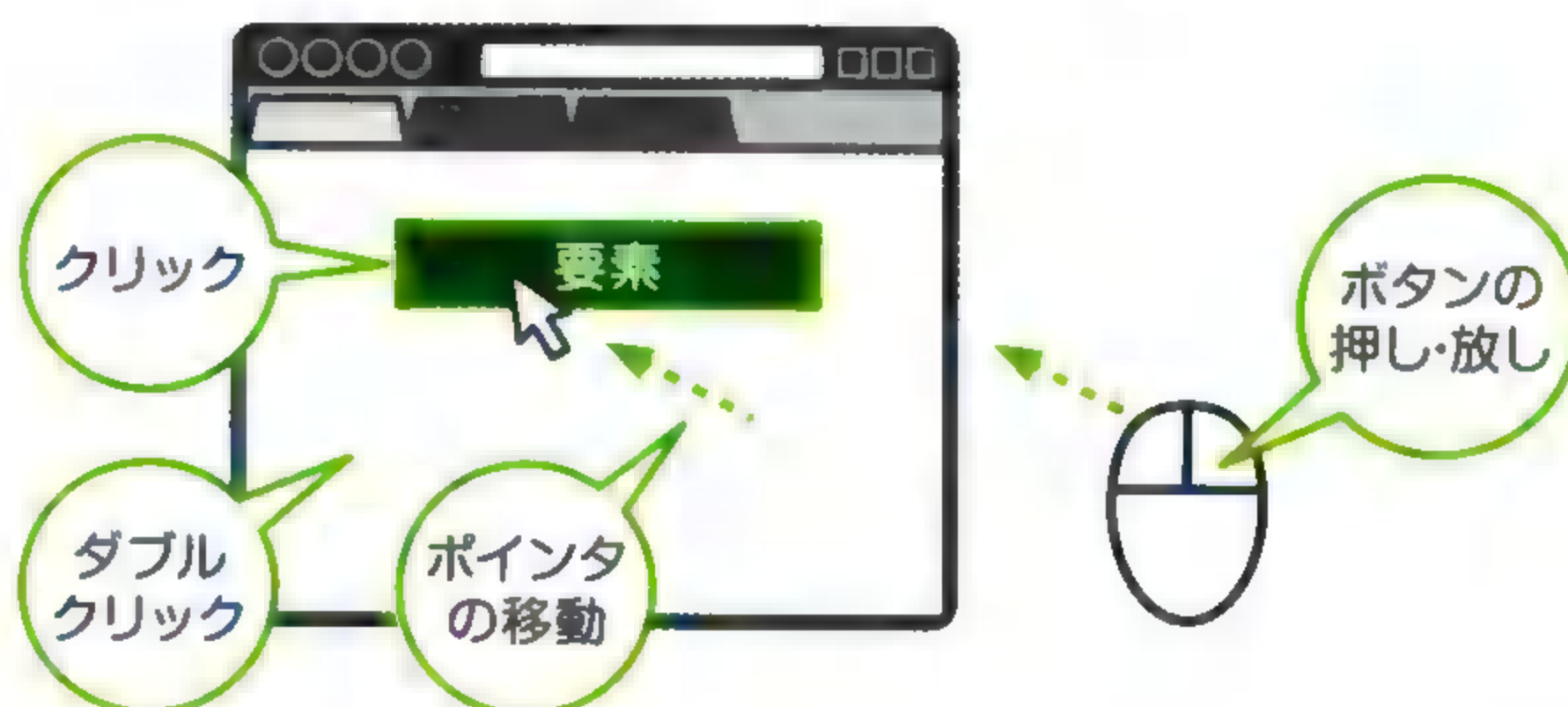
SECTION

1

JavaScript のプログラムではよく使われる、イベントの基本について説明します。

● イベントとは？

イベントとは、ブラウザ上でユーザーが行うさまざまな動作のことです。たとえば、ブラウザ画面上でのマウスのクリックやダブルクリック、マウスポインタの移動、マウスボタンの押し下げ・放しなどがイベントになります。



イベントにはイベント名が付けられています。おもなイベント名は以下のとおりです。

▼ おもなイベント

イベント名	いつ発生するか？
load	読み込みが終わったとき
click	マウスをクリックしたとき
dblclick	マウスをダブルクリックしたとき
mouseover	マウスポインタが要素に重なったとき
mouseout	マウスポインタが要素から外れたとき
mousedown	マウスボタンを押したとき
mousemove	マウスを移動したとき
mouseup	マウスボタンを放したとき

MEMO

この他にもイベントはたくさんあります。

JavaScript を使うと、イベントの発生をきっかけにして、何らかの処理を実行させることができます。この処理のことを、**イベントハンドラ**といいます。

Lesson 3



イベントハンドラの使い方

イベントハンドラを使ったプログラムを作るために必要な、基本的な考え方を学びます。

● イベントハンドラを使うには？

イベントは、HTML 文書の要素単位で発生します。イベントハンドラを使ったプログラムを作るにあたって、考えなくてはならないポイントが3つあります。「要素」と「イベント」と「イベントハンドラ」です。

- ① どの要素に？
- ② どのイベントが発生したときに？
- ③ どのイベントハンドラを実行するか？

たとえば、h1 要素がクリックされたときに、警告ダイアログボックスに「イベントが発生した！」という文字列が表示されるようなサンプルプログラムを作ってみたいとします。3つのポイントはこのようになります。

- ① h1 要素に（要素）
- ② クリックイベントが発生したとき（イベント）
- ③ 警告ダイアログボックスを表示する（イベントハンドラ）

その上で、対象となる「要素」と「イベント」と「イベントハンドラ」を関連付ける方法は、おもに2つあります。開始タグの中で関連付ける方法と、プロパティを使って関連付ける方法です。

順番に見ていきましょう。

Lesson 3



開始タグの中で 関連付ける方法

イベントとイベントハンドラの関連付けを、開始タグの中の属性を使って行う方法を説明します。

● 開始タグの中で関連付けるには？

1つ目は、対象となる要素の開始タグの中に書く方法です。

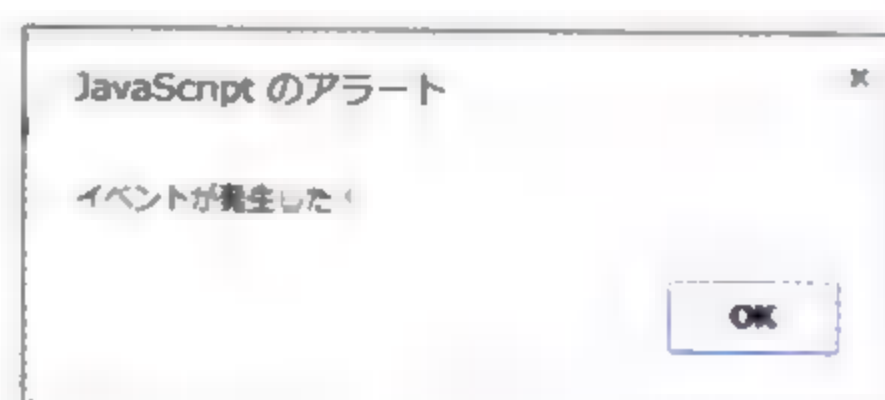
▼ 開始タグの中に書く方法

<要素名 on イベント名 =" イベントハンドラ "> ~ </要素名>

今回の場合、h1 要素の開始タグの中に、このように書きます。

```
1 <!DOCTYPE html>
2 <html lang="ja">
3 <head>
4   <meta charset="utf-8">
5   <title> プログラム実行テスト </title>
6 </head>
7 <body>
8   <h1 onclick="window.alert(' イベントが発生した! ');"> プログ
   ラム実行テスト </h1>
9 </body>
10 </html>
```

プログラムを実行して、h1 要素をクリックすると、警告ダイアログボックスが表示されます。



h1 要素をクリックすると
警告ダイアログが表示される

onclickの部分を書き換えれば、他のイベントでも応用可能です。

mouseover イベントの場合

```
8      <h1 onmouseover="window.alert(' イベントが発生した！ ');"> プ  
      プログラム実行テスト </h1>
```

dblclick イベントの場合

```
8      <h1 ondblclick="window.alert(' イベントが発生した！ ');"> プ  
      グラム実行テスト </h1>
```

● イベントハンドラを関数にするには？

今回のイベントハンドラを関数にすることもできます。関数にするには、このように書きます。

```
1      <!DOCTYPE html>  
2      <html lang="ja">  
3      <head>  
4          <meta charset="utf-8">  
5          <title> プログラム実行テスト </title>  
6          <script>  
7              function strClick() {  
8                  window.alert(' イベントが発生した！ ');  
9              }  
10         </script>  
11     </head>  
12     <body>  
13         <h1 onclick="strClick()"> プログラム実行テスト </h1>  
14     </body>  
15 </html>
```

script要素の中で定義したstrClick()関数を、h1要素のonclick属性の値に指定して呼び出しています。

プロパティを使って 関連付ける方法

SECTION

4

イベントとイベントハンドラの関連付けを、プロパティを使って行う方法を説明します。

● プロパティを使って関連付けるには？

2つ目の方法は、プロパティを使う方法です。イベント対象となる要素を `getElementById()` メソッドで取得して変数に代入します。そして、そのプロパティとしてイベントを指定し、プロパティの値としてイベントハンドラを指定するという方法です。

▼ プロパティで関連付ける方法

```
オブジェクト名.onイベント名 = function() {  
    イベントハンドラ  
};
```

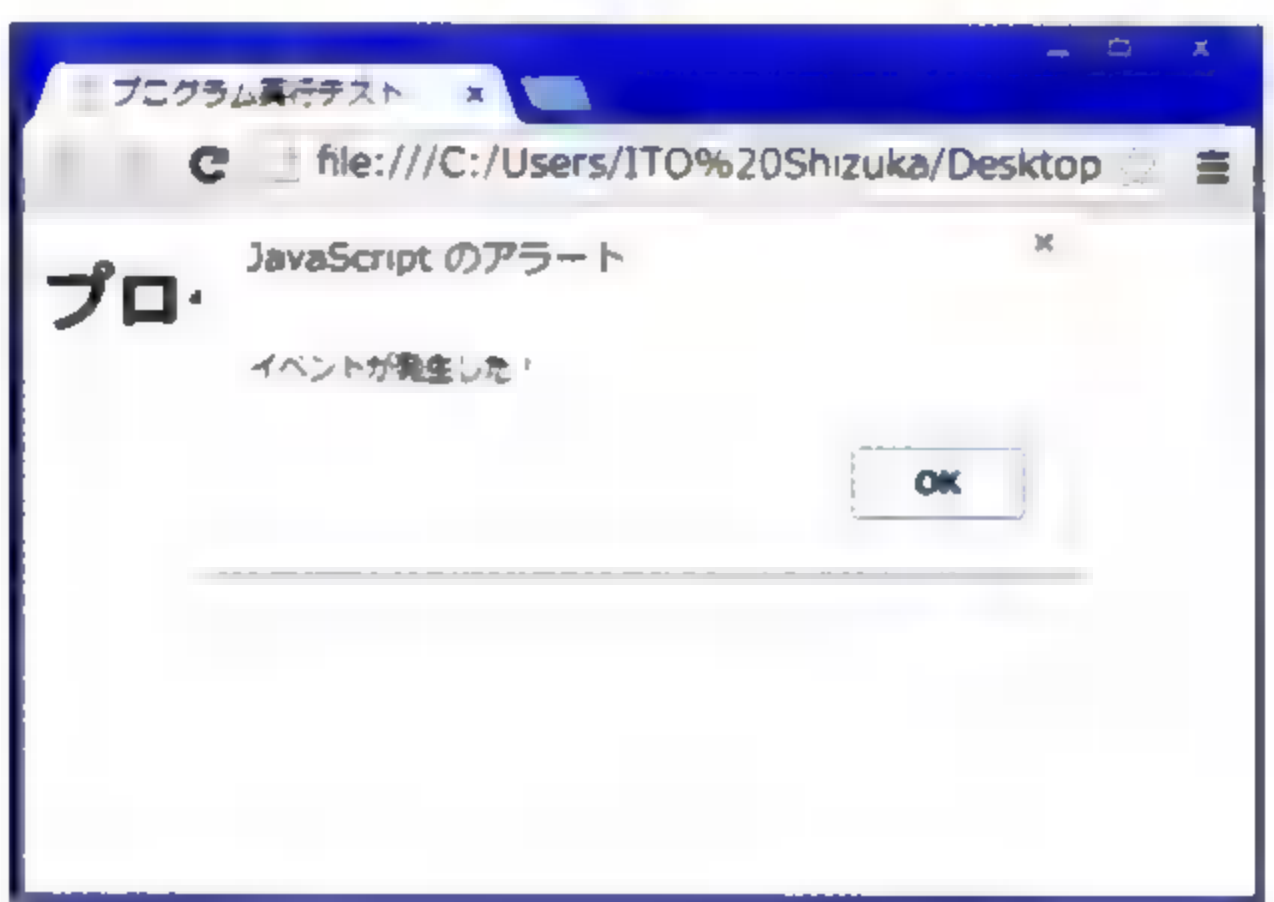
この方法を使ったサンプルプログラムは、以下のとおりです。

```
3 <head>  
4   <meta charset="utf-8">  
5   <title> プログラム実行テスト </title>  
6 </head>  
7 <body>  
8   <h1 id="header"> プログラム実行テスト </h1>  
9   <script>  
10    var e = document.getElementById('header');  
11    e.onclick = function() {  
12      window.alert(' イベントが発生した！ ');  
13    };  
14  </script>  
15 </body>
```




まず、getElementById()メソッドでh1要素をオブジェクトとして取得し、変数eに代入します(10行目)。そして、変数eのonclickプロパティに、関数としてイベントハンドラを代入します(11～13行目)。

プログラムを実行して、h1要素をクリックすると、警告ダイアログボックスが表示されます。



h1要素をクリックすると
警告ダイアログが表示される

10～13行目の処理は、変数を使わず、このように書いてもかまいません。

```
10      document.getElementById('header').onclick =
11      function() {
12          window.alert(' イベントが発生した! ');
13      };
```

● イベント名はすべて小文字で書く

プロパティとしてイベント名を使うときは、必ずすべて小文字で書かなくてははいけません。

- e.onclick = function() {
- × e.onCClick = function() {
- document.getElementById('header').onclick =
- × document.getElementById('header').onCClick =

大文字を混ぜて書いてしまうと、プログラムがエラーになり、実行されません。注意してください。



load イベントの 使い方

SECTION

5

3-3-1 で紹介したイベントの中から、他のイベントとは異なる使われ方をする load イベントについて説明します。

● load イベントとは？

load イベントは、関連付けられた要素を読み込み終わったときに発生するイベントです。body 要素なら body 要素が、img 要素なら img 要素を読み込み終わったときに発生します。

load イベントにイベントハンドラを設定する場合ですが、body 要素に設定するときは、`window.onload` プロパティに設定します。これは load イベントだけでなく、他のイベントも同様です。

▼ 構文：body 要素の load イベントにイベントハンドラを設定

```
window.onload = function(){ イベントハンドラ };
```

その他の要素に設定する場合は、取得した要素オブジェクトの `onload` プロパティに設定します。これは 2-3-4 で紹介した方法と同じです。

▼ 構文：その他の要素の load イベントにイベントハンドラを設定

```
要素オブジェクト名.onload = function(){ イベントハンドラ };
```

たとえば、body 要素が読み込まれたときに警告ダイアログボックスが表示されるようなサンプルプログラムは、このようになります。

```
3 <head>
4   <meta charset="utf-8">
5   <title> プログラム実行テスト </title>
6   <script>
7     window.onload = function(){
```



```

8         alert(' イベントが発生した！ ');
9     };
10    </script>
11    </head>
12    <body>
13        <h1> プログラム実行テスト </h1>
14    </body>

```

● window.onload の特殊な使い方

これまでのレッスンで紹介してきたDOMやイベントハンドラは、じつは設置する場所によっては、きちんと実行されないことがあります。

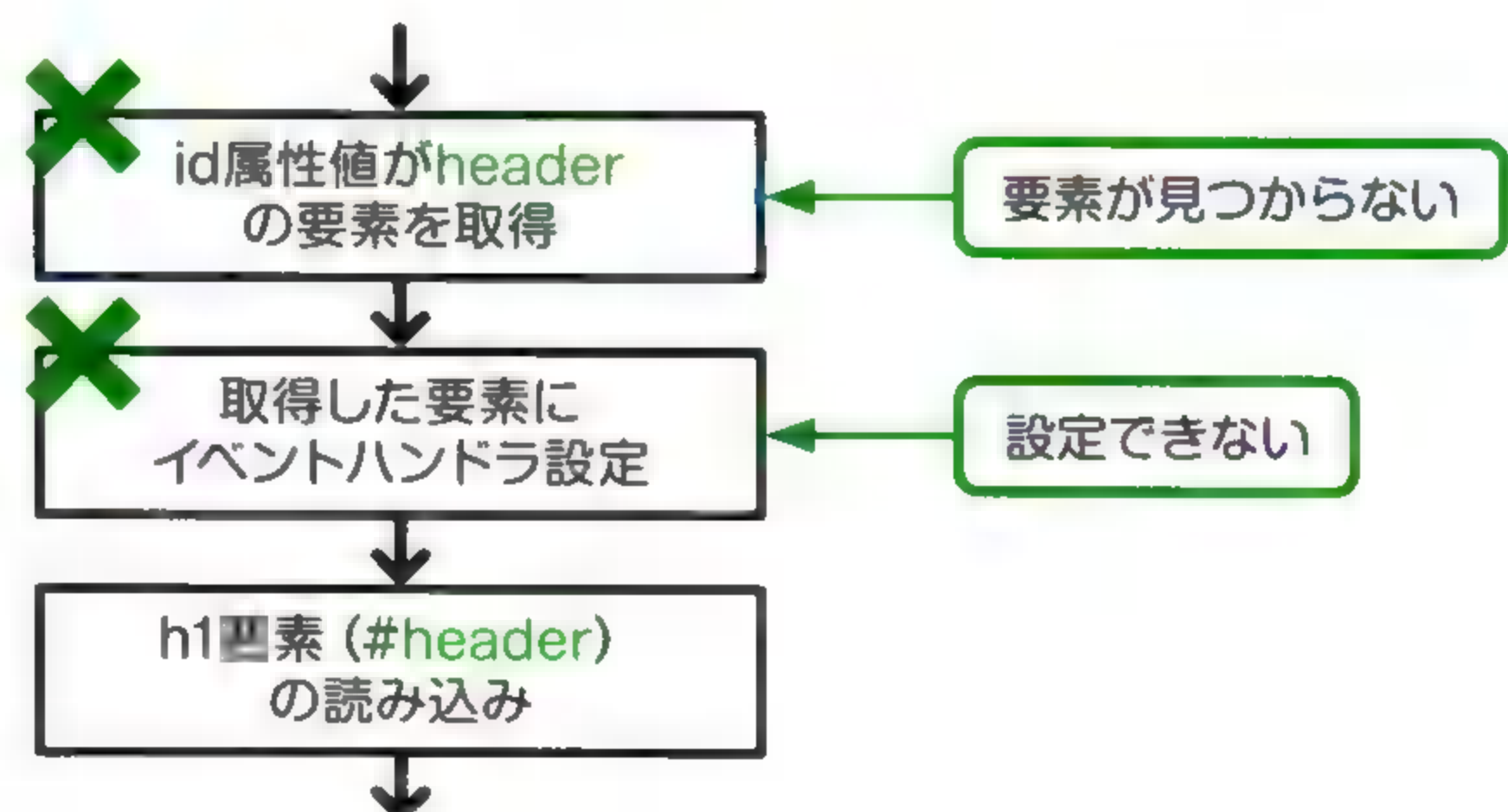
たとえば、3-3-4で紹介したサンプルですが、以下のように、script要素をh1要素よりも前に書くと、クリックしてもイベントハンドラは実行されません。

```

7    <body>
8        <script>
9            var e = document.getElementById('header');
10           e.onclick = function() {
11               window.alert(' イベントが発生した！ ');
12           };
13        </script>
14        <h1 id="header"> プログラム実行テスト </h1>
15    </body>

```

この原因は、1-3-1で説明したプログラムの実行順です。プログラムは通常、先頭から順番に実行されます。このプログラムの場合、9行目でid属性値がheaderである要素を取得しようとしています。その時点でまだh1要素は読み込まれていません。つまり、対象となる要素がないため、結果として、イベントハンドラが設定されないのです。



書く場所によって実行が左右されないようにするには、DOMやイベントハンドラのソースコードを、まるごとwindow.onloadのイベントハンドラとして設定するという方法があります。具体的には、9行目と14行目に、以下のような記述を挿入します。

```
7  <body>
8    <script>
9      window.onload = function(){
10        var e = document.getElementById('header');
11        e.onclick = function() {
12          window.alert(' イベントが発生した！ ');
13        };
14      };
15    </script>
16    <h1 id="header"> プログラム実行テスト </h1>
17  </body>
```

このように書くと、10～13行目のソースコードは、必ずbody要素内のすべての要素が読み込み終わってから実行されるようになります。

つまり、script要素をHTML文書内のどこに配置しても、要素を取得でき、イベントハンドラを設定することができるようになります。

というわけで、DOMを使った処理を書く場合は、HTML文書のどこに書いてもいいですが、処理を丸ごとwindow.onloadプロパティのイベントハンドラとして設定するようにしてください。



addEventListener() メソッド

1つのイベントに複数のイベントハンドラを設定できる
addEventListener() メソッドについて説明します。



● addEventListener() メソッドの使い方

これまでに紹介してきた2つの方法では、1つの要素や1つのイベントに対して、1つのイベントハンドラしか設定できません。

1つのイベントに複数のイベントハンドラを設定したいときには、addEventListener (アドイベントリスナー) メソッドを使います。

▼ 構文: addEventListener() メソッド

要素名.**addEventListener**(' イベント名 ', function(){
 イベントハンドラ
}, **false**)

たとえば、h1 要素のクリックイベントに、警告ダイアログボックスを表示するイベントハンドラと、h1 要素の文字の色を赤にするイベントハンドラを設定するには、以下のように書きます。

```
9      window.onload = function(){
10          var e = document.getElementById('header');
11          e.addEventListener('click', function() {
12              window.alert(' イベントが発生した! ');
13          }, false);
14          e.addEventListener('click', function() {
15              e.style.color = 'red';
16          }, false);
17      };
```



タッチデバイスのイベント

SECTION

7

スマートフォンやタブレット端末などのタッチデバイスでは、画面上の要素を指で操作するため、マウスイベントの代わりに、タッチイベントが使われます。

● タッチイベントとは？

スマートフォンやタブレット端末などのタッチデバイスの操作には、マウスは使いません。よって、マウス関連のイベントであるmouseover、mouseout、mousedown、mousemove、mouseupは使うことができません。

その代わりに用意されているのが、タッチイベントです。

タッチイベントには、touchstart、touchmove、touchendの3つがあります。



touchstart



touchmove



touchend

▼ タッチイベント

イベント名	内容
touchstart	タッチが始まったとき
touchmove	タッチしたまま移動したとき
touchend	タッチが終わったとき

タッチイベントでは、タッチしている指の数も認識することができます。タッチイベントを使えば、指を2本使った動作であるピンチインやピンチアウトなどの動きに対応したプログラムも作れます。

Day 3 Lesson 4

jQuery の使い方

このレッスンでは、jQueryの基本的な使い方を学びます。

- 1 jQueryとは何か？
- 2 jQueryをインストールする
- 3 jQueryを読み込む設定をする
- 4 jQueryの記述方法
- 5 css() メソッドで要素の色を変える
- 6 html() メソッドで要素の中身を変える
- 7 fadeOut() メソッドで
アニメーション効果
- 8 メソッドを繋げて書くメソッドチェーン
- 9 イベント処理を設定する



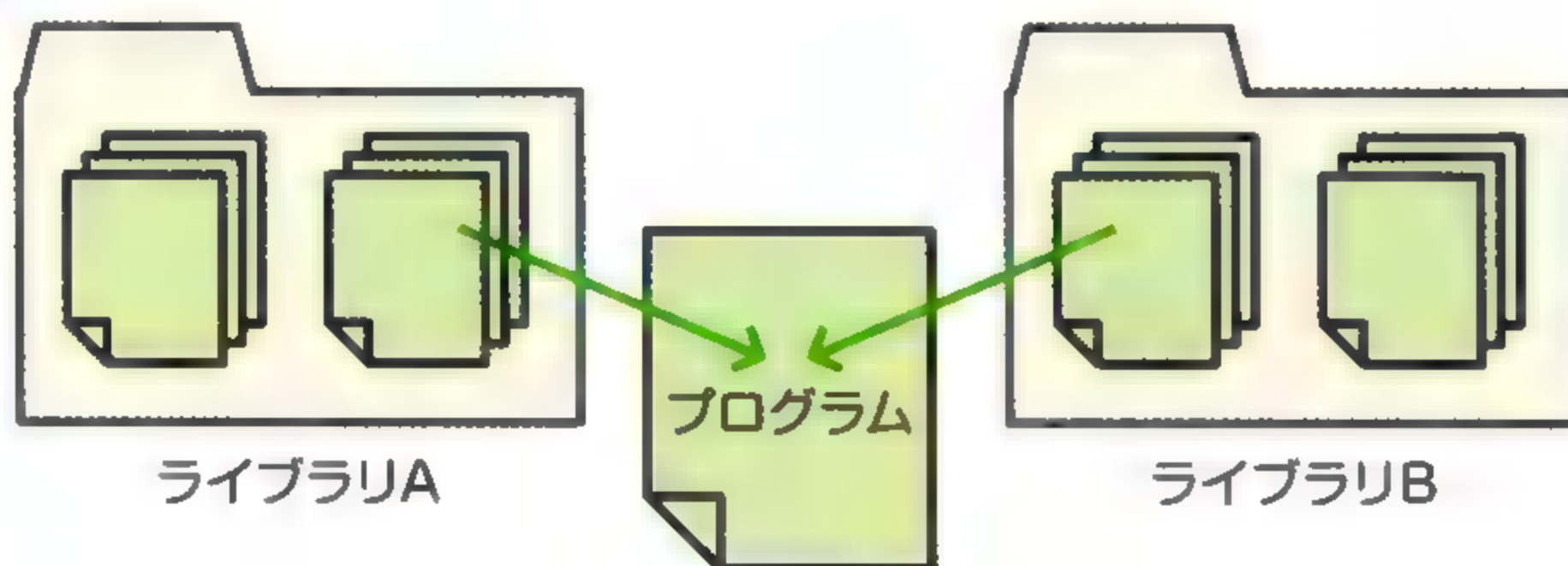
jQuery とは何か？

ここ最近、JavaScript でプログラムを作るときによく使われている jQuery とは何なのかを説明します。

● jQuery とは？

jQuery とは、JavaScript のライブラリの1つです。ライブラリ(図書館)というのは、まさにその名前のとおり、プログラマにとっての図書館のようなもので、先輩プログラマが、いろいろなプログラムで共通して使われることの多い処理を切り出して、あらかじめまとめておいてくれる、プログラムパーツの集合体のことです。おかげで、誰でも、いつでも、無料で、便利な機能を使えるようになっています。

▼ ライブラリはプログラムパーツの集合体



ライブラリを使うと、プログラムをゼロから作るよりも、手間をかけずに、短時間で作ることができます。また、同じような処理を、何度も繰り返し書く必要がありません。これから学ぶ jQuery は、DOM 操作関連の機能を集めたライブラリです。

MEMO

JavaScript のライブラリには、jQuery の他にもいろいろあります。プログラム設計のためのフレームワーク (AngularJS など)、DOM 操作 (圧倒的に jQuery が使われている)、スマホサイト制作 (jQuery Mobile など)、グラフィック関連 (three.js、D3.js など) などがあります。

Lesson 4



jQuery をインストールする

jQuery を使うには、jQuery ファイルをダウンロードして、適切なフォルダに保存する必要があります。

Lesson 4



jQuery をインストールする

● jQuery のインストール方法

jQuery のファイルを、公式サイトからダウンロードして、保存します。



- ① 検索サイトで「jquery」と検索するなどして、「<http://jquery.com>」を開く



- ② 「Download」ボタンをクリック

MEMO

サイトのデザインが変わっている場合は、「Download」というボタンや文字を探してクリックしてください。



- ③ 画面をスクロールして、**最新バージョン**を探す

MEMO

執筆時点の最新バージョンは2.1.1です。もしもっと新しいものがあれば、以下「2.1.1」の部分は、最新版の数字に読み替えてください。



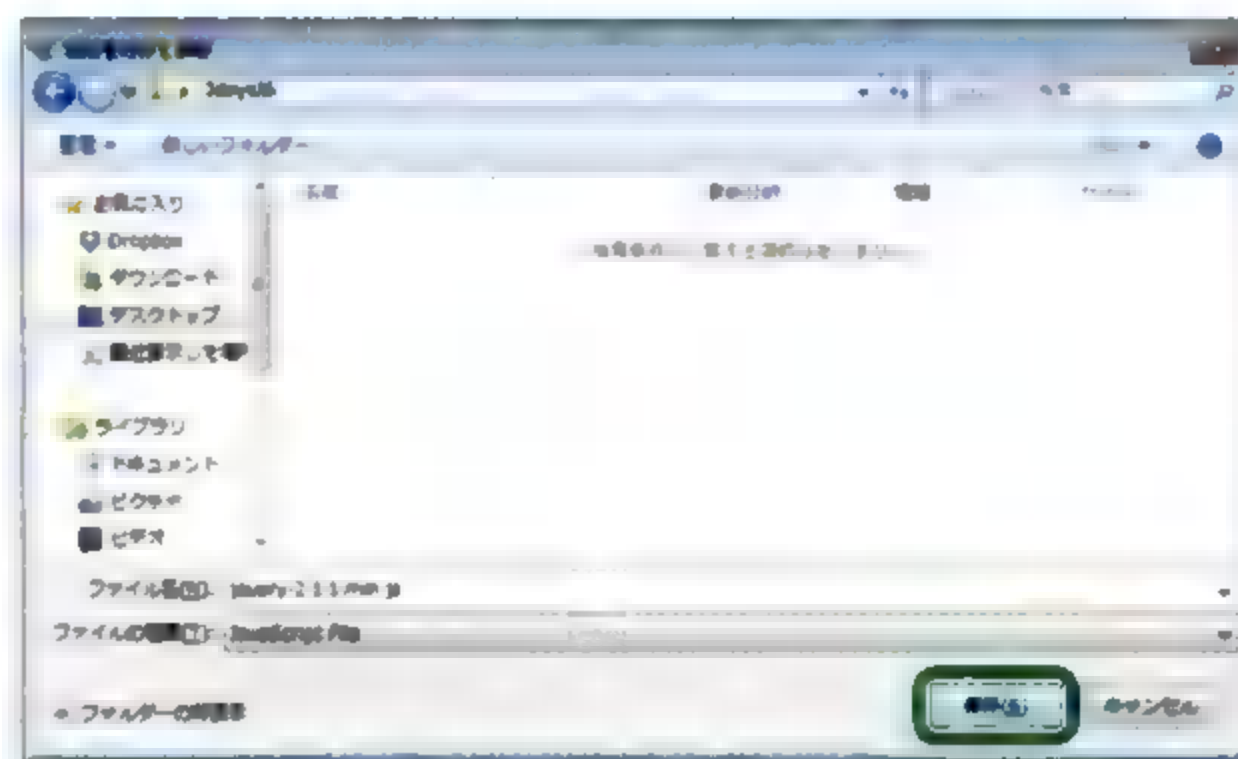
④ 「Download the compressed, production jQuery 2.1.1」をクリック

MEMO

「compressed（圧縮）」は、ソースコードから不要なスペースや改行を取り除いて、ファイルサイズを小さくしてあるファイルです。



⑤ Ctrlキー+「S」キーを押して、ファイル保存ダイアログを表示



⑥ sample.htmlと同じフォルダに保存

MEMO

ファイル名の拡張子が「.js」、ファイルの種類が「JavaScript File」になっているか確認してください。



⑦ 保存したファイルの名前を「jquery.js」に変更

jQueryファイルのインストールはこれで終わりです。

Lesson 4



jQuery を読み込む設定をする

jQuery ファイルを読み込むための行を書き加えます。

● jQuery を読み込む設定

まず、jQueryがHTMLファイルに読み込まれるように設定します。head要素のtitle要素のすぐあとに、次の1行を書き加えてください。

```
1 <!DOCTYPE html>
2 <html lang="ja">
3 <head>
4   <meta charset="utf-8">
5   <title> プログラム実行テスト </title>
6   <script src="jquery.js"></script>
7 </head>
```

これでHTMLファイルがブラウザに読み込まれたときに、同時にjQueryのファイルも読み込まれます。

HTMLファイルにJavaScriptのソースコードを直接記述する場合は、jQuery読み込みの行よりもあとに別途script要素を作り、その中に記述してください。

もし、JavaScriptの外部ファイル(.js)を読み込む場合も、読み込むためのscript要素は、jQuery読み込みの行よりもあとに書いてください。

```
<script src="jquery.js"></script>
<script src="sample.js"></script> // 外部 JS ファイルの読み込み
<script>
  // ここに処理を記述する
</script>
```





jQuery の記述方法

jQuery でソースコードを書く方法を説明します。

● jQuery のソースコードを書くには？

jQuery の基本的な操作は、2段階の手順をとります。

- ① HTML の要素を取得する
- ② 取得した要素を操作する

この2つの手順を、次のように1行で書くのがもっとも基本的な書き方です。

▼ 構文：jQuery の基本的な書き方

```
$ ( 取得したい要素 ). メソッド名 ( 引数1, 引数2, ... );
```

取得された要素は、**jQuery オブジェクト**という種類のオブジェクトになります。jQuery オブジェクトには、いろいろなメソッドが用意されています。

● 取得したい要素の指定方法

「取得したい要素」を指定するには、**CSS セレクタ**を使います。よく使われるCSS セレクタは以下の3種類です。

セレクタ名	取得できる要素	書き方
タイプセレクタ	指定した要素	'要素名'
クラスセレクタ	指定した class 属性値を持つ要素	'.class 属性値'
ID セレクタ	指定した id 属性値を持つ要素	'#id 属性値'

以下のようなソースコードを例に、具体的に説明します。

```

1  <!DOCTYPE html>
2  <html lang="ja">
3  <head>
4    <meta charset="utf-8">
5    <title> プログラム実行テスト </title>
6    <script src="jquery.js"></script>
7    <script>
8      // ここに jQuery のソースコードを記述する
9    </script>
10 </head>
11 <body>
12   <h1> プログラム実行テスト </h1>
13   <p id="top"> 第 1 段落 </p>
14   <p class="contents"> 第 2 段落 </p>
15   <p id="bottom"> 第 3 段落 </p>
16 </body>
17 </html>

```

タイプセクタを使って、h1 要素 (12 行目) を取得するには、8 行目にこのように書きます。

```

8  $('h1');

```

次に、クラスセクタを使って2番目のp要素 (14 行目) を取得するには、このように書きます。

```

8  $('.contents');

```

最後に、IDセクタを使って1番目のp要素 (13 行目) を取得するには、このように書きます。

```

8  $('#top');

```



CSS() メソッドで要素の色を変える

jQuery オブジェクトの `css()` メソッドを使って、要素の色を変える方法を説明します。

● 要素の色を変えるには？

前のセクションでは、ただjQueryオブジェクトとして取得しただけなので、ブラウザ画面上では何の変化もありません。jQueryオブジェクトの**`css()`メソッド**というものを使って、取得したh1要素の色を赤に変えてみましょう。

`css()`メソッドで要素の色を変えるには、このように書きます。

▼ 構文：`css()`メソッド

```
$(セレクト).css('color', '色')
```

先ほど取得したh1要素の色を赤に変えるには、このように書きます。

```
8 | $('h1').css('color', 'red');
```

これで実行してみると、h1要素の色が赤に変わる……かと思いきや、黒のままになっているかと思います。読み込んだ要素に対して何らかの操作を加えるには、じつはもう1つ書かなければならないことがあります。それが**`$(function(){})`**です。

● `$(function(){})` の使い方

前のレッスンで、イベントハンドラを実行するには、HTML文書がすべて読み込み終わるのを待ってからでないときちんと動作しないので、`window.onload` イベントを使う方法を説明しました。

jQueryにも同じ問題があります。HTML文書がすべて読み込み終わってからでないと、要素の取得ができないので、jQueryの操作が実行されないのです。

jQueryでは、\$(function(){})という書き方をします。

▼ \$(function(){}) の構文

```
$(function() {
    jQueryの処理
});
```

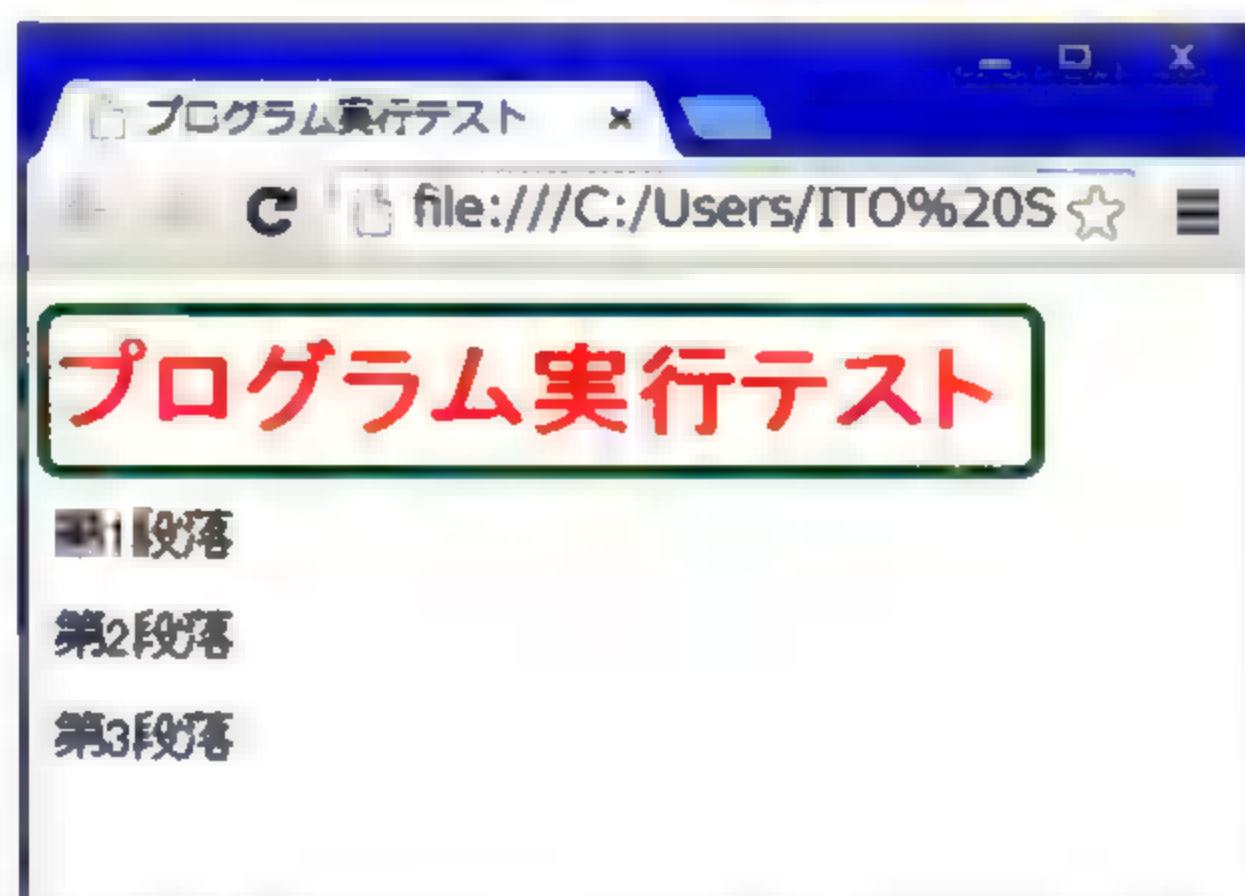
jQueryの処理は、この構文の中カッコ{}の中に書きます。こうすることで、HTML文書の読み込みが完了してから、jQueryの処理が実行されるようになります。先ほどのjQueryは、以下のように書きます。

```
8 $(function() {
9     $('h1').css('color', 'red');
10 });
```

MEMO

カッコの数が多いので、書き間違い、書き忘れをしないように気をつけてください。

これを実行すると、h1要素の色が赤に変わります。



h1要素の色が変更された

MEMO

色を指定する第2引数は、16進数のカラーコードを使って書いても大丈夫です。たとえば、赤なら#ff0000となります。



html() メソッドで要素の中身を変える

jQuery オブジェクトの html() メソッドを使って、要素の中身を変える方法を説明します。

●要素の中身を変えるには？

取得した要素の中身を変えるには、**html()メソッド**を使います。html()メソッドはこうのように書きます。

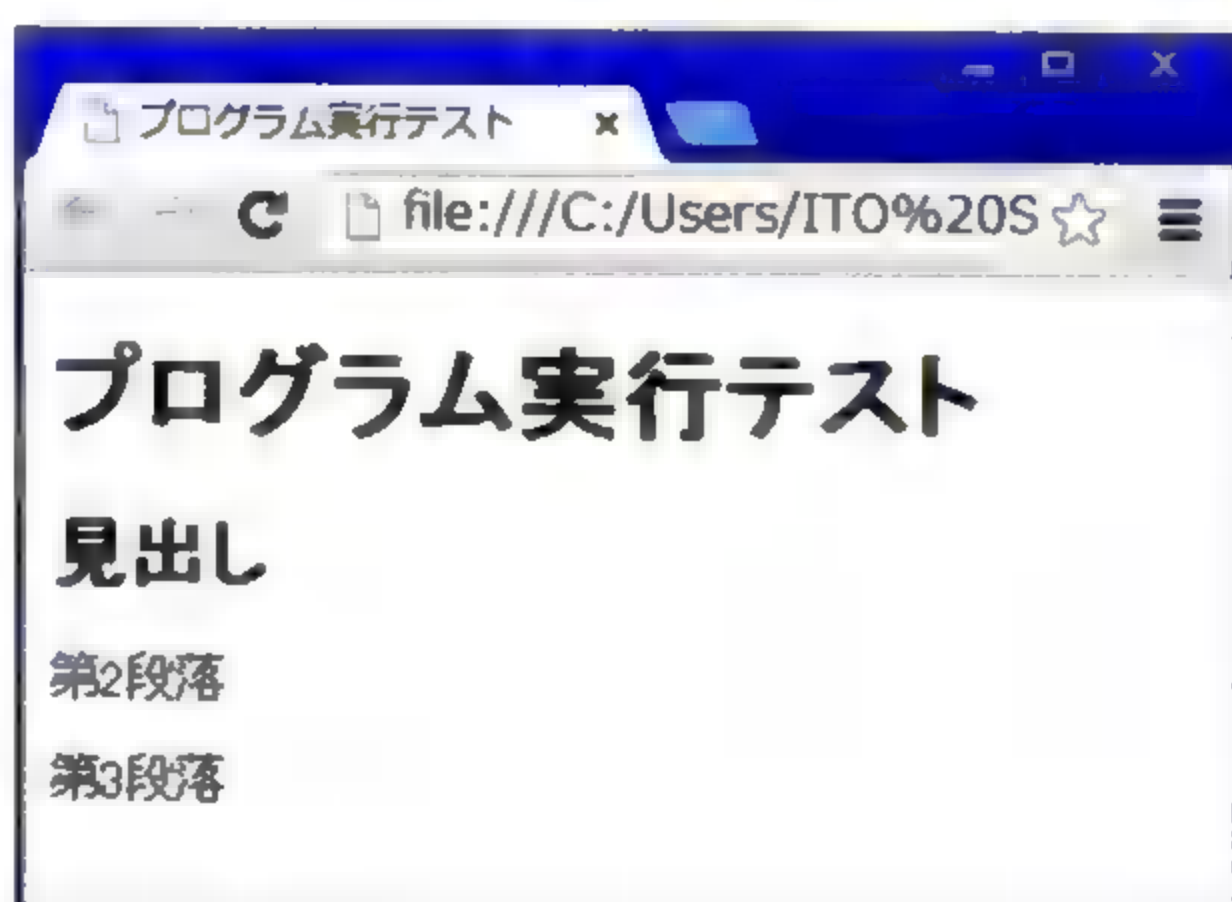
▼ 構文: html() メソッド

```
$(セレクト).html('変更後のHTML')
```

先ほどと同じサンプルプログラムで、IDセクタ (#top) を使って1番目のp要素を取得し、h2要素に変更してみましょう。このように書きます。

```
8  $(function() {  
9    $('#top').html('<h2>見出し</h2>');  
10 });
```

プログラムを実行してみて、最初のp要素(第1段落)がh2要素の「見出し」に変わっていれば成功です。



p要素がh2要素に変更された

Lesson 4



fadeOut() メソッドでアニメーション効果

jQuery オブジェクトの fadeOut() メソッドを使って、要素がフェードアウトして見えなくなるようにするアニメーション効果の書き方を説明します。

●要素をフェードアウトさせるには？

取得した要素がフェードアウトして見えなくなるようなアニメーションにするには、fadeOut() メソッドを使います。fadeOut() メソッドはこのように書きます。

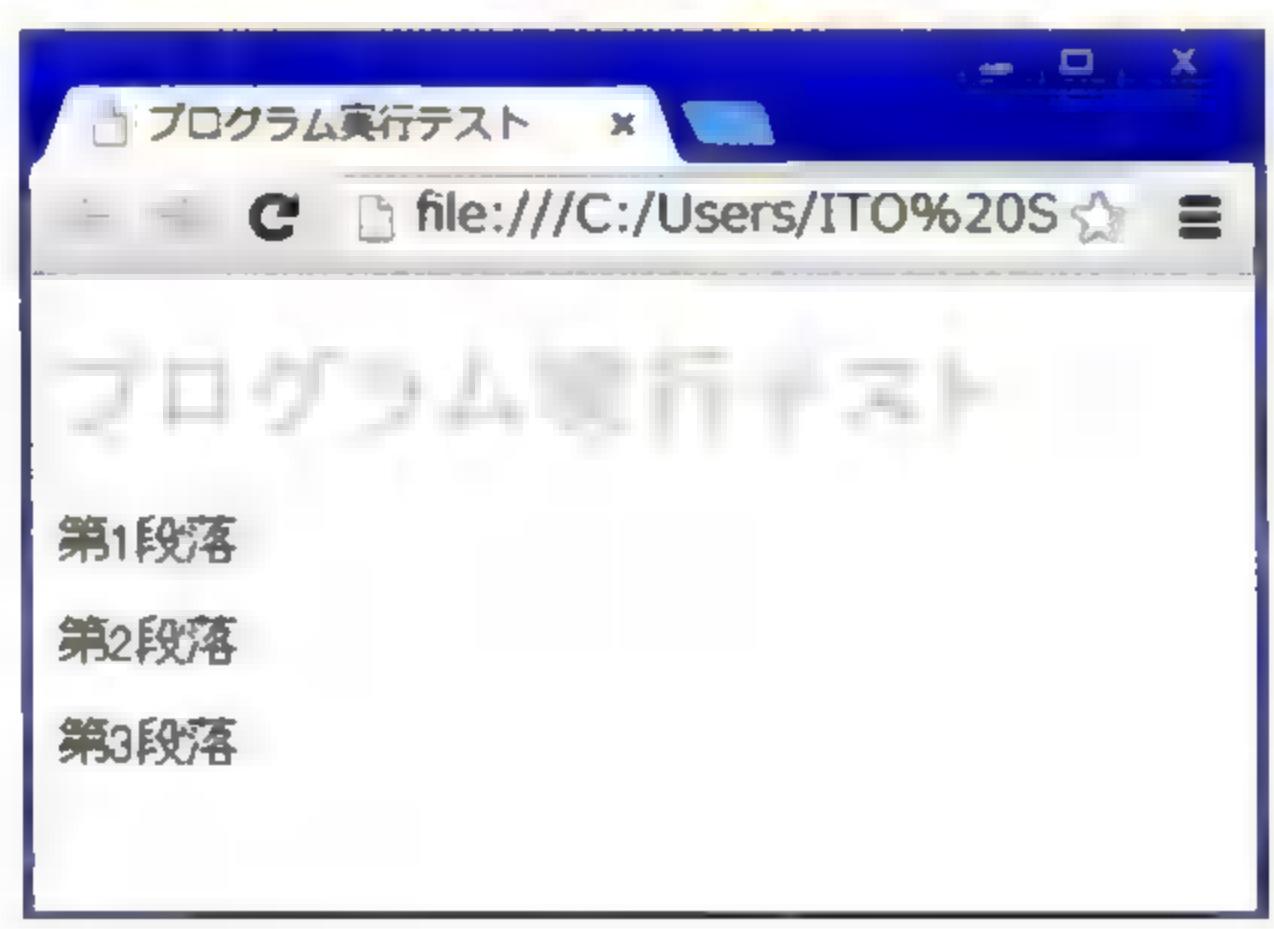
▼ 構文：fadeOut() メソッド

\$(セレクト).fadeOut()

先ほどと同じサンプルプログラムで、h1 要素を取得し、フェードアウトさせるには、このように書きます。

```
8 $(function() {
9   $('h1').fadeOut();
10 });
```

プログラムを実行して、h1 要素がフェードアウトして見えなくなれば成功です。



h1 要素がフェードアウトする

MEMO

fadeOut() メソッドの「O」は大文字です。小文字で書くとエラーになって実行されません。また、カッコの中に数値（ミリ秒単位）を書くと、見えなくなるまでの時間を指定することができます。初期値は400（ミリ秒）です。



メソッドを繋げて書く メソッドチェーン

jQuery オブジェクトのメソッドを複数書くときの書き方を説明します。

●メソッドを複数設定するには？

1つのjQueryオブジェクトに複数のメソッドを指定するときには、メソッドをピリオド(.)で繋げて書きます。この書き方を**メソッドチェーン**といいます。

▼ 構文：メソッドチェーンの書き方

`$(セレクト).メソッド名1().メソッド名2()`

先ほどと同じサンプルプログラムで、h1要素を取得し、いったんフェードアウトさせてからフェードインさせてみましょう。フェードインさせるには、`fadeIn()`メソッドを使います。

```
8  $(function() {  
9    $('h1').fadeOut().fadeIn();  
10 });
```

プログラムを実行してみてください。h1要素がいったんフェードアウトして見えなくなってから、フェードインして表示されれば成功です。メソッドがたくさん繋がると、1行が長くなって、ソースコードが分かりにくくなることがあります。そんなときは、メソッドの前で改行して、それぞれインデントすると見やすくなります。

```
8  $(function() {  
9    $('h1')  
10      .fadeOut()  
11      .fadeIn();  
12 });
```




Lesson 4



イベント処理を設定する

click イベントを例にして、jQuery を使ったイベント処理の書き方を説明します。

● イベントとイベントハンドラを関連付けるには？

jQuery でもイベントとイベントハンドラを関連付ける処理を書くことができます。

▼ 構文：イベントとイベントハンドラの関連付け

```
$(セレクト).イベント名(function() {  
    イベントハンドラ  
});
```

またカッコをたくさん使いますので、書くときには気をつけてください。

たとえば、h1 要素がクリックされたら、イベントハンドラが実行されるような処理を書いてみましょう。イベントハンドラは、前のセクションで使った h1 要素の色を赤に変える処理にしてみます。script 要素の中に、このように書きます。

```
8   $(function(){  
9       $('h1').click(function() {  
10          $('h1').css('color', 'red');  
11      });  
12  });
```

実行してみてください。表示された h1 要素をクリックして、赤に変われば成功です。

MEMO

イベントハンドラは jQuery でなくても OK です。たとえば、alert() メソッドの処理を書けば、クリックすると警告ダイアログボックスが表示されます。



索引

- **英数字**
 - \$ 166
 - \$(function({})) 168
- **A**
 - addEventListener() メソッド 159
 - alert() メソッド 128
 - appendChild() メソッド 147
 - Array オブジェクト 11
- **B**
 - Boolean オブジェクト 123
 - break 文 77
- **C**
 - confirm() メソッド 128
 - createElement() メソッド 147
 - css() メソッド 168
 - CSS セレクタ 166
- **D**
 - Date オブジェクト 108
 - default 文 78
 - do ~ while 文 88
 - document オブジェクト 137
 - DOM 140
- **E**
 - Error オブジェクト 123
- **F**
 - fadeOut() メソッド 172
 - fadeOut() メソッド 171
 - form オブジェクト 138
 - for 文 86
 - Function オブジェクト 122
- **G**
 - getAttribute() メソッド 147
 - getElementById() メソッド 146
 - getElementsByClassName() メソッド 148
 - getElementsByTagName() メソッド 141
- **H**
 - history オブジェクト 135
 - html() メソッド 170
 - HTML 文書 137
- **I**
 - ID セレクタ 166
 - if ~ else if ~ else 文 74
 - if ~ else 文 72
 - if 文 70
 - image オブジェクト 138
 - innerHTML プロパティ 143
- **J**
 - jQuery 162
 - js ファイル 24
- **L**
 - length プロパティ (history) 135
 - length プロパティ (String) 119
 - length プロパティ (配列) 60
 - link オブジェクト 138
 - load イベント 156
 - location.href プロパティ 134
 - location オブジェクト 133
- **M**
 - Math オブジェクト 111
- **N**
 - navigator オブジェクト 136
 - new 演算子 107
 - null 68
 - Number オブジェクト 121
- **O**
 - Object オブジェクト 123
 - open() メソッド 129
- **P**
 - pop() メソッド 57
 - prompt() メソッド 128
 - push() メソッド 58
- **Q**
 - querySelector() メソッド 147
 - querySelectorAll() メソッド 147
- **R**
 - RegExp オブジェクト 113
 - return 文 97
- **S**
 - setAttribute() メソッド 147
 - shift() メソッド 55
 - splice() メソッド 59
 - String オブジェクト 119
 - switch 文 77
- **U**
 - undefined 68
 - unshift() メソッド 56
- **W**
 - Web ブラウザ 14
 - while 文 82
 - window.onload 157
 - window オブジェクト 126
- **あ**
 - アルゴリズム 32
- **い**
 - イベント 150
 - イベントハンドラ 150
 - インクリメント演算子 92
 - インスタンス 107
 - インデックス 51
 - インデント 36
- **え**
 - エスケープシーケンス 43
- **お**
 - オブジェクト 62
 - オブジェクトツリー 65
 - オブジェクト名 65

● か			
カウンタ (変数)	83		
返り値	98		
拡張子	27		
仮引数	100		
関数	94		
関数リテラル	103		
● き			
基本データ型	48		
基本のオブジェクト	106		
局所変数	102		
● く			
組み込みオブジェクト ..	106		
クラスセクタ	166		
繰り返し文	34		
グローバル変数	102		
● こ			
コメント	47		
固有のオブジェクト	106		
コンストラクタ	107		
● さ			
再読込	26		
算術演算子	45		
参照	40		
● し			
条件式	79		
条件文	33		
条件分岐	33		
初期化	39		
真偽値	44		
真理値	44		
● す			
数値リテラル	41		
● せ			
正規表現	113		
正規表現リテラル	115		
制御文	33		
セミコロン	36		
宣言	38		
選択	33		
		添え字	51
● た			
大域変数	102		
代入	39		
代入演算子	39		
タイプセクタ	166		
タッチイベント	160		
単純データ型	48		
● て			
テキストエディタ	14		
デクリメント演算子	92		
● ね			
ネイティブオブジェクト	106		
● は			
配列	51		
配列名	54		
配列リテラル	118		
パラメータ	100		
反復	34		
● ひ			
比較演算子	79		
引数	99		
ビルトインオブジェクト	106		
● ふ			
プール値	44		
複合データ型	48		
プライマリデータ型	48		
ブラウザ	14		
ブラウザオブジェクト ..	126		
プリミティブデータ型 ..	48		
ブロック	71		
プロパティ	62		
● へ			
変数	38		
● む			
無限ループ	85		
無名関数	103		
● め			
メソッド	62, 104		
● も			
文字化け	21		
文字列リテラル	42		
戻り値	96		
● よ			
要素 (配列)	51		
要素 (HTML)	141		
呼び出し (関数)	95		
● ら行			
ライブラリ	162		
リロード	26		
連結演算子	46		
ローカル変数	102		
論理演算子	80		

ブック・キャラクターデザイン 坂本 真一郎 (クオルデザイン)

本文 DTP・イラスト作成 西嶋 正

編集協力 内藤 貴志

著者紹介 伊藤 静香

テクニカルライター。

著書に『1 週間で基本情報技術者の基礎が学べる本』

『アルゴリズムを、はじめよう』。

3日でマスター JavaScript

2014 年 8 月 1 日 初版第 1 刷発行

著 者 伊藤 静香

発行人 片柳 秀夫

編集人 佐藤 英一

発行所 ソシム株式会社

<http://www.socym.co.jp/>

〒101-0064 東京都千代田区猿楽町 1-5-15

猿楽町 SS ビル

TEL 03-5217-2400 (代表)

FAX 03-5217-2420

印刷・製本 シナノ印刷株式会社

定価はカバーに表示しております。

落丁・乱丁は弊社編集部までお送りください。送料弊社負担にてお取り替えいたします。

ISBN978-4-88337-934-7 Printed in JAPAN

©2014 ITO Shizuka. All rights reserved.

- 本書の一部または全部について、個人で使用するほかは、著作権上、著者およびソシム株式会社の承諾を得ずに無断で複写／複製することは禁じられております。
- 本書の内容の運用によって、いかなる障害が生じても、ソシム株式会社、著者のいずれも責任を負いかねますのであらかじめご了承ください。
- 本書の内容に関して、ご質問やご意見などがございましたら、上記のソシムの Web サイトの「お問い合わせ」よりご連絡ください。なお、電話によるお問い合わせ、本書の内容を超えたご質問には応じられませんのでご了承ください。



ISBN978-4-88337-934-7

C2055 ¥1800E

定価：本体 1800円(税別)



9784883379347



1922055018009

CONTENTS

Day 1

- ① プログラム学習の準備
- ② 1行だけのプログラムを作ってみる
- ③ プログラム作成前に必要な知識
- ④ 変数のしくみと単純なデータ
- ⑤ 配列
- ⑥ オブジェクト

Day 2

- ① 条件文
- ② 繰り返し文
- ③ 関数
- ④ 組み込みオブジェクト

Day 3

- ① Windowオブジェクト
- ② DOM
- ③ イベントとイベントハンドラ
- ④ jQuery